데이터 베이스

도로 싱크홀 및 복구 상황 발생시 투입인력, 필요자재 데이터베이스

데이터 베이스 설계 배경

중앙일보 2025.03.25. 17:47

지난 24일 오후 서울 강동구 명일동 한영외고 앞 도로에서 발생한 싱크홀(땅꺼짐)로 추락한 남성 박모(34)씨가 심정지 상태로 발견됐다. 싱크홀 발생 18시간 만이다.

소방 당국은 25일 브리핑을 열고 “이날 오전 11시 22분쯤 실종자를 발견했고, 오후 12시 36분쯤 구조를 완료했다”고 했다.

이어 “호흡과 의식이 없는 심정지 상태로 발견됐다”고 밝혔다. 박씨는 인근 병원으로 이송돼 최종 사망 판정을 받았다.

지자체 인프라 도로 싱크홀 복구 담당 공무원을 위한
데이터 분석용 데이터베이스를 설계

데이터 베이스 설계

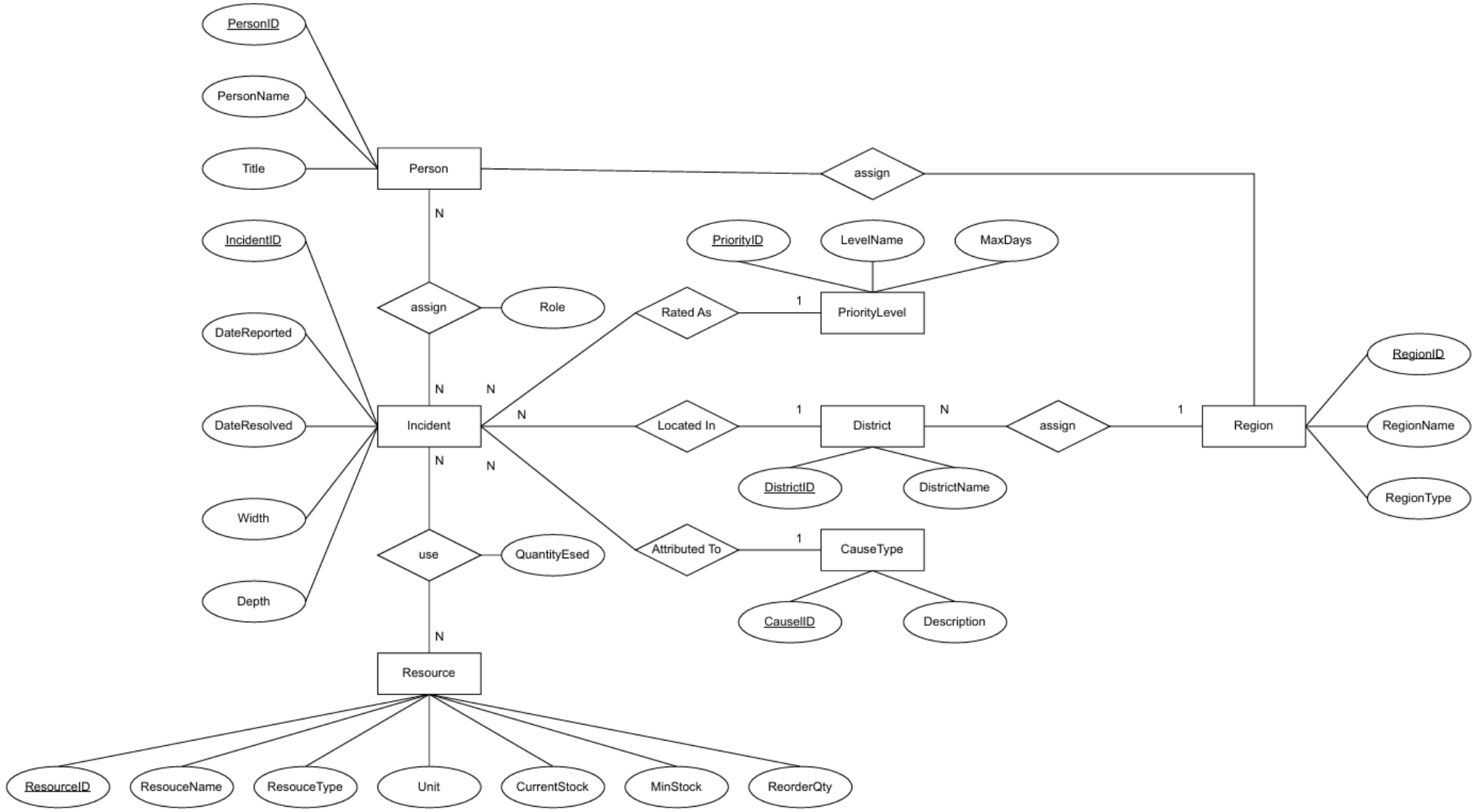
현장 판단과 의사결정에 필요한 정보를 신속히 얻는 것이 목표

싱크홀을 예방하기 위해서는
체계적인 사고 기록 관리와 통계 분석이 필수
*MySQL 관계형 데이터베이스

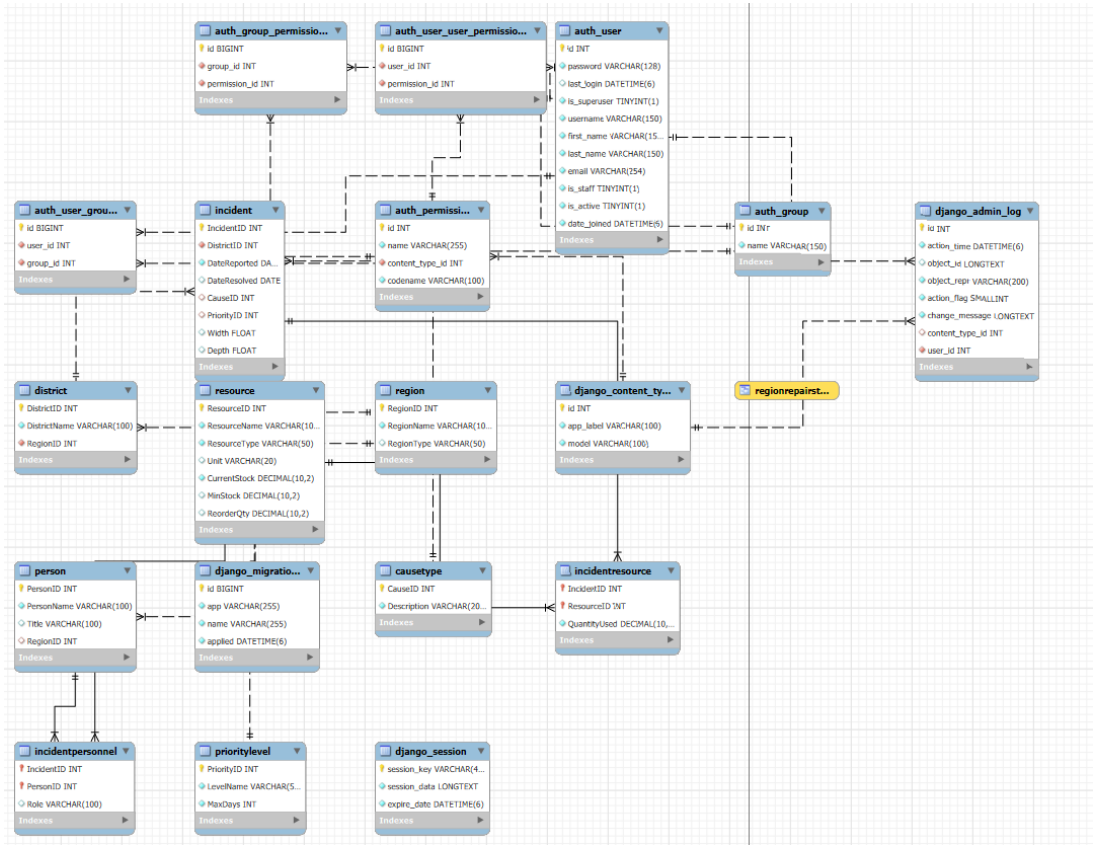
- 자재 사용량 통계
- 사고 별 복구 소요일
- 자재 재고 임계치 도달 여부

싱크홀이라는 사고에 대응할 수 있게 MYSQL데이터베이스를 설계

ER DIAGRAM


PersonID
PersonName
Title
Person
assign
N
IncidentID
PriorityID
LevelName
MaxDays
assign
Role
Rated As
1
PriorityLevel
DateReported
N
N
N
N
DateResolved
Incident
Located In
1
District
N
assign
1
Region
RegionID
RegionName
RegionType
DistrictID
DistrictName
N
N
Width
use
QuantityEsed
Attributed To
1
CauseType
Depth
CauseID
Description
N
Resource
ResourceID
ResouceName
ResouceType
Unit
CurrentStock
MinStock
ReorderQty

IE DIAGRAM



MySQL Workbench

어느 지역에서 사고가 많이 발생하는지,
어떤 원인과 어떤 우선순위 사고가 잦은지,
누가(인력), 무엇을(자원) 투입했는지를 파악할 수 있게 9개의 테이블을 설계하고 이를 활용할 수 있는 트랜잭션과 프로시져, 뷰를 설계하였습니다.

Djnago - python

장고를 활용하여 mysql과 연결해 장고를 사용한 데이터베이스 조작 및 활용을 간단하게 구현해보았습니다.

Mysqlclient를 활용하였으며 매핑과 CRUD를 사용할 수 있는 코드베이스를 설계하고 구현하였습니다.

TABLE

Region		
RegionID (PK)	INT	광역 ID
RegionName	VARCHAR(100)	시·도명
RegionType	VARCHAR(50)	‘Metropolitan City’ 등

지자체 행정구역의 상위 개념 한 Region은 여러 District를 포함합니다. 수도와 시,도 명을 파악할 수 있습니다.

District		
DistrictID (PK)	INT	시·군·구 ID
DistrictName	VARCHAR(100)	시·군·구명
RegionID (FK)	INT	소속 Region

사고 발생 지역 단위 (예: 구/군). 각 District는 어느 한 Region에 속하며 여러 Incident를 가집니다 (Region:District = 1:N 관계)

PriorityLevel		
PriorityID (PK)	INT	우선순위 ID
LevelName	VARCHAR(50)	‘High’·‘Medium’·‘Low’
MaxDays	INT	허용 복구일수

사고 중요도 또는 우선순위 등급 (*긴급*, *보통*, *낮음*). 각 등급별로 허용 최대 복구일 수를 정의해 두면, 이를 초과하는 사고에 지연 경보를 줄 수 있습니다.

CauseType		
CauseID (PK)	INT	원인 ID
Description	VARCHAR(200)	예: ‘노후 배수관 파손’

싱크홀 발생 원인 (상수관 파열, 노후 하수관 붕괴, 지하수유형 침식 등). 통계 분석 시 사용되며, 하나의 CauseType에 여러 Incident가 연관될 수 있습니다.

TABLE

Person		
PersonID	INT	PK
PersonName	VARCHAR(100)	NOT NULL
Title	VARCHAR(100)	
RegionID	INT	FK → Region(RegionID)

복구 작업에 쓰이는 자재 또는 장비 목록 ,재고관리용 속성으로 현재고와 **최소 필요재고**, 권장 보충량을 저장합니다. 담당자는 이 정보를 통해 자재 부족을 미리 파악하고 보충할 수 있습니다.

Incident		
IncidentID	INT	PK
DistrictID	INT	NOT NULL, FK → District(DistrictID)
DateReported	DATE	NOT NULL
DateResolved	DATE	
CauseID	INT	FK → CauseType(CauseID)
PriorityID	INT	FK → PriorityLevel(PriorityID)
Width	FLOAT	싱크홀 폭 (m)
Depth	FLOAT	싱크홀 깊이 (m)

Resource		
ResourceID	INT	PK
ResourceName	VARCHAR(100)	NOT NULL
ResourceType	VARCHAR(50)	NOT NULL (‘Material’ 등)
Unit	VARCHAR(20)	(‘ton’, ‘pcs’ 등)
CurrentStock	DECIMAL(10,2)	NOT NULL
MinStock	DECIMAL(10,2)	(임계 재고량)
ReorderQty	DECIMAL(10,2)	(재주문 권장량)

Resource

싱크홀 사고 발생 및 복구 기록 테이블로, 본 시스템의 중심 엔티티입니다. 주요 속성으로 사고 ID, 발생지역, 발생일, 복구완료일, 원인, 우선순위등을 갖고 있습니다.

또한 싱크홀 크기 등의 정보를 위해 폭과 깊이필드를 포함했습니다.

Incident

복구 작업 관련 담당자(공무원 등) 정보. 여러 사건을 담당할 수 있으며,
Region를 속성으로 가져 어떤 지역 담당자인지 표시합니다
한 Incident에 여러 사람이 투입될 수 있어 다대다 관계를 가집니다.

TABLE

IncidentPersonnel		
IncidentID	INT	PK (복합), FK → Incident(IncidentID) ON DELETE CASCADE
PersonID	INT	PK (복합), FK → Person(PersonID)
Role	VARCHAR(100)	

사고-담당자의 교차 테이블로, 한 Incident에 여러 Person 투입을 기록합니다.
PK는 (IncidentID, PersonID)이며 두 칼럼 모두 FK로 Incident와 Person을 참조합니다.

필요시 해당 사고 내 역할(Role) 필드를 기록할 수 있습니다.
이를 통해 사고당 평균 투입 인원, 개별 직원의 사고 처리 건수 등의 분석이 가능합니다

IncidentResource		
IncidentID	INT	PK (복합), FK → Incident(IncidentID) ON DELETE CASCADE
ResourceID	INT	PK (복합), FK → Resource(ResourceID)
QuantityUsed	DECIMAL(10,2)	사용 수량

총 9개의 테이블을 설계 MYSQL DDL로 구현하였음

데이터 살펴보기 – 생성형 AI로 데이터 생성

GPT를 사용해 테이블에 맞는 가상의 데이터를 생성하였음을 밝힙니다.

RegionID	RegionName	RegionType
1	Seoul	Metropolitan City
2	Busan	Metropolitan City
3	Daegu	Metropolitan City

*Region table

DistrictID	DistrictName	RegionID
101	Gangnam-gu	1
102	Jongno-gu	1
201	Busanjin-gu	2

*District table

CauseID	Description
1	Water pipe burst
2	Sewer collapse
3	Heavy rain erosion

*CauseType table

PriorityID	LevelName	MaxDays
1	High	30
2	Normal	90
3	Low	180

*PriorityLevel

PersonID	PersonName	Title	RegionID
1	Lee Alpha	Engineer	1
2	Kim Beta	Inspector	1
3	Park Gamma	Field Lead	2

*Resource table

데이터 살펴보기 – 생성형 AI로 데이터 생성

GPT를 사용해 테이블에 맞는 가상의 데이터를 생성하였음을 밝힙니다.

IncidentID	DistrictID	DateReported	DateResolved	CauseID	PriorityID	Width	Depth
201	201	2024-09-21	2024-09-25	8	1	10.0	8.0
202	201	2024-08-21	2025-02-01	1	2	5.0	3.0
203	102	2025-05-15	2025-05-16	3	1	2.0	2.0

*Incident table

IncidentID	PersonID	Role
201	4	NULL
201	5	NULL
201	6	NULL

*IncidentPersonnel table

IncidentID	ResourceID	QuantityUsed
201	1	220.0
201	2	50.0
201	3	20.0

* IncidentResource table

PersonID	PersonName	Title	RegionID
1	Lee Alpha	Engineer	1
2	Kim Beta	Inspector	1
3	Park Gamma	Field Lead	2

* Person table

VIEW 지역별 복구 실적 요약 뷰

```
CREATE VIEW RegionRepairStats AS
SELECT r.RegionName AS Region,
       AVG(DATEDIFF(i.DateResolved, i.DateReported)) AS AvgRepairDays,
       AVG(ip.PersonCount) AS AvgPersonnel,
       AVG(ir.ResourceCount) AS AvgResourceTypes

FROM Incident i
JOIN District d   ON i.DistrictID = d.DistrictID
JOIN Region r     ON d.RegionID = r.RegionID

LEFT JOIN (
    SELECT IncidentID, COUNT(*) AS PersonCount
    FROM IncidentPersonnel
    GROUP BY IncidentID
) ip ON i.IncidentID = ip.IncidentID

LEFT JOIN (
    SELECT IncidentID, COUNT(*) AS ResourceCount
    FROM IncidentResource
    GROUP BY IncidentID
) ir ON i.IncidentID = ir.IncidentID

WHERE i.DateResolved IS NOT NULL
GROUP BY r.RegionID, r.RegionName;
```

Region별로 평균 복구일수, 평균 투입 인원, 평균 투입 자재 종류 수를 보여줍니다.

뷰 정의에 집계함수와 GROUP BY를 활용했으며,

서브쿼리로 Incident당 인원수와 자재종류 개수를 계산하여 조인한 뒤 Region별로 묶었습니다

어느 지역의 복구에 시간이 오래 걸리는지, 인력이 많이 드는지 등을 한눈에 파악할 수 있습니다.

TRANSACTION

미해결 사고 조회 TRANSACTION 읽기

```
START TRANSACTION;

SELECT COUNT(*) AS OpenIncidents
FROM Incident
WHERE DateResolved IS NULL;

SELECT IncidentID, DistrictID, DateReported, PriorityID
FROM Incident
WHERE DateResolved IS NULL;
COMMIT;
```

현재 복구 완료되지 않은 사고들의 현황을 파악하는
읽기전용 트랜잭션입니다.

먼저 미해결 사고 건수를 집계하고,
이어서 그 상세 목록을 같은 시점 기준으로 조회합니다.

TRANSACTION

신규 사고 등록 TRANSACTION 생성

```
START TRANSACTION;

INSERT INTO Incident
(IncidentID, DistrictID, DateReported, DateResolved, CauseID, PriorityID, Width, Depth)
VALUES (300, 102, CURDATE(), NULL, 3, 1, 1.5, 2.0);

INSERT INTO IncidentPersonnel VALUES (300, 1, NULL);
INSERT INTO IncidentPersonnel VALUES (300, 2, NULL);

INSERT INTO IncidentResource VALUES (300, 7, 1.0);
INSERT INTO IncidentResource VALUES (300, 8, 4.0);

COMMIT;
```

새로운 싱크홀 사고 발생을 기록하고, 동시에 초기 대응 조치로
투입된 인력과 자재도 함께 등록하는 복합 삽입입니다.

TRANSACTION

자재 사용 수정 TRANSACTION 업데이트

```
START TRANSACTION;
SET @oldQty = (SELECT QuantityUsed FROM IncidentResource
               WHERE IncidentID = 201 AND ResourceID = 1);

UPDATE IncidentResource
SET QuantityUsed = 220.0

WHERE IncidentID = 201 AND ResourceID = 1;
UPDATE Resource
SET CurrentStock = CurrentStock - (220.0 - @oldQty)
WHERE ResourceID = 1;
COMMIT;
```

사고 처리 중 잘못 기록된 자재 사용량을 정정하고,
그에 따라 재고를 보정하는 트랜잭션입니다.

IncidentResource의 QuantityUsed를 수정하면서,
Resource 테이블의 CurrentStock도 연동하여 수정해야
두 테이블의 일관성이 유지됩니다.

TRANSACTION

오기록 삭제 TRANSACTION 삭제

```
START TRANSACTION;

SET @qty := (SELECT QuantityUsed FROM IncidentResource
             WHERE IncidentID = 209 AND ResourceID = 6);
DELETE FROM IncidentResource
WHERE IncidentID = 209 AND ResourceID = 6;

UPDATE Resource
SET CurrentStock = CurrentStock + @qty
WHERE ResourceID = 6;
COMMIT;
```

잘못 입력된 자재 투입 기록을 삭제하고,
실제로는 사용되지 않은 자재를 재고에 환원하는 트랜잭션입니다.

어떤 사고에 특정 자재 사용이 잘못 기록되었다면,
IncidentResource 레코드를 삭제하고 그 수량만큼
Resource 재고를 되돌립니다.

PROCEDURE

저장 PROCEDURE

```
DELIMITER $$
CREATE PROCEDURE RestockMaterial
(IN p_resourceID INT, IN p_addAmount DECIMAL(10,2))
BEGIN
UPDATE Resource
    SET CurrentStock = CurrentStock + p_addAmount
    WHERE ResourceID = p_resourceID;
SELECT ResourceID, ResourceName, CurrentStock
    FROM Resource
    WHERE ResourceID = p_resourceID;
END $$
DELIMITER ;
```

```
CALL RestockMaterial(8, 20.0);
```

자재 보충작업을 자동화하는 프로시저

RestockMaterial(p_resourceID, p_addAmount)는
입력한 자재 ID의 재고를 지정한 수량만큼 증가시키고,업데이트 후 새로운 재고량을 반환합니다.

MYSQL 데이터베이스 활용

데이터베이스 활용 SQL문 예시

Cause	IncidentCount	AvgDaysToFix
Natural karst	1	4.0000
Water pipe burst	4	90.7500
Heavy rain erosion	2	7.5000
Underground cavity	1	10.0000

* 사고복구평균일수

ResourceName	CurrentStock	MinStock
Asphalt	30.00	50.00

* 임계치 미만자재

IncidentID	ResourceTypesUsed
201	5
202	5
203	4
204	3
205	2
206	3
207	4
208	2
209	3
210	4
300	2

* 자재별 사용량

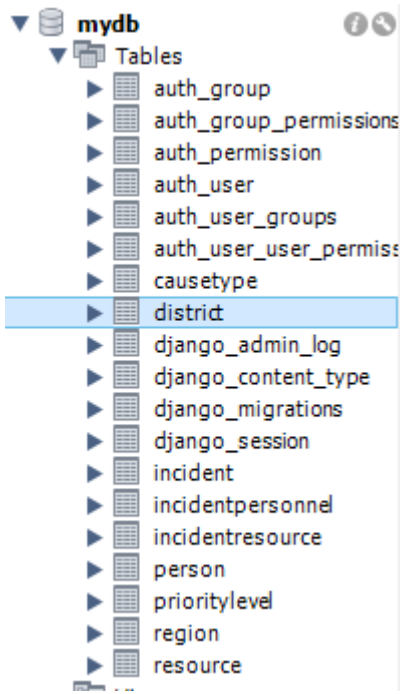
YearMonth	ResourceName	TotalUsed
2025-01	Steel Beam	5.00
2025-01	Traffic Cone	5.00
2025-03	Concrete	5.00
2025-03	Gravel/Sand	30.00
2025-03	Traffic Cone	2.00
2025-04	Fill Soil	150.00
2025-04	Rebar	20.00
2025-04	Traffic Cone	5.00
2025-05	Concrete	5.00
2025-05	Fill Soil	20.00
2025-05	Steel Plate	1.00
2025-05	Traffic Cone	1.00
2025-06	Asphalt	11.00
2025-06	Fill Soil	55.00
2025-06	Steel Plate	3.00
2025-06	Traffic Cone	8.00

* 월별 자재 사용량

YearMonth	ResourceName	TotalUsed
2024-08	Asphalt	20.00
2024-08	Concrete	10.00
2024-08	Fill Soil	100.00
2024-08	Steel Plate	1.00
2024-08	Traffic Cone	2.00
2024-09	Asphalt	50.00
2024-09	Concrete	20.00
2024-09	Fill Soil	220.00
2024-09	Steel Plate	1.00
2024-09	Traffic Cone	4.00
2024-10	Asphalt	10.00
2024-10	Fill Soil	30.00
2024-10	Traffic Cone	3.00
2025-01	Asphalt	30.00
2025-01	Fill Soil	200.00
2025-01	Steel Beam	5.00

* 월별 자재 사용량2

Django - MySql



자재 사용량 통계

1) 사고별 자재 사용량

사고 ID	발생일	위치	총 자재량 (kg)
1	2025-06-10	서울시 강남구 역삼동	120
2	2025-06-15	부산시 해운대구 반여동	80
3	2025-05-28	대구시 중구 동성로	50

2) 월별 자재 사용량 및 사고 건수

월	사고 건수	총 자재량 (kg)
2025년 5월	1	50
2025년 6월	2	200

*데이터베이스와 장고를 연결하여 간단한 crud 시스템 설계 및 구현

Github :https://github.com/wlrma0108/sinkhole_mysql.git

관리자용 페이지로 간단한 통계를 볼 수 있고 django의 기능을 활용한 CRUD가 가능하다.

데이터 삽입 조회 SQL

TableName	RowCount
Resource	10
Region	10
PriorityLevel	3
Person	10
IncidentResource	35
IncidentPersonnel	20
Incident	10
District	10
CauseType	10

*삽입한 데이터를 count를 통해서 확인

IncidentID	DistrictID	DateReported	PriorityID
207	601	2025-06-10	2
208	801	2024-01-15	3
NULL	NULL	NULL	NULL

* 미해결 사고 조회 TRANSACTION 확인

IncidentID	ResourceID	ResourceName	QuantityUsed
300	7	Steel Plate	1.00
300	8	Traffic Cone	4.00

* 신규 사고 등록 TRANSACTION 확인 – 추가 sql쿼리 사용

IncidentID	ResourceID	QuantityUsed
201	1	220.00
NULL	NULL	NULL

* 자재 사용 수정 TRANSACTION 확인 Quantuse 220

ResourceID	ResourceName	CurrentStock
1	Fill Soil	500.00
NULL	NULL	NULL

* 자재 사용 수정 TRANSACTION 확인 Currentstock 500

ResourceID	ResourceName	CurrentStock
8	Traffic Cone	120.00

* Procedure CALL 확인

IncidentID	ResourceID	QuantityUsed
NULL	NULL	NULL

* 오기록 삭제 TRANSACTION 확인 조회되는 행없음

ResourceID	ResourceName	CurrentStock
6	Steel Beam	150.00
NULL	NULL	NULL

* 오기록 삭제 TRANSACTION 확인 current 수 증가 확인가능

TEAM 춘식

김 호 중

Django, cloud, k8s, mlops
Git hub : @wlrma0108
기여도: 25%

허 윤 팀장

소프트웨어학부 21학번
Git hub : @HeoYun36
기여도: 25%

최 태 민

소프트웨어학부 21학번
Git hub : @v5ql
기여도: 25%

김 채 연

소프트웨어학부 22학번
Git hub : @erummohagi
기여도: 25%

데이터 베이스

발표를 들어 주셔서 감사합니다.

TEAM 춘식