

Two thin, dark blue diagonal lines are present on the left side of the slide. One line starts near the top left and extends towards the top right. The other line starts near the bottom left and extends towards the middle right.

99. Project Guideline

Project Guideline

■ 프로젝트 주제 선정

➤ 부적합:

- 단순한 앱 (TODO, Memo 등) → Entity, Relationship, Transaction 등이 빈약함
- 정적 콘텐츠 중심의 블로그 (댓글/좋아요 기능 X) → DB 연동 중심이 아닌 Front-end 중심
- 불법/민감 데이터 수집/적용 (의료, 금융 소스 이용) → 윤리·법적 리스크
- NoSQL 사용 → 과목 범위/목표와 불일치 (RDB와 병행은 OK)
- 광범위한 데이터 (전세계 항공사 실시간 데이터) → 수집·정합성·범위 관리 실패 위험 큼

➤ 적합:

- 학사관리: 수강·성적·상담 예약 등 transaction이나 역할기반접근제어 (RBAC)가 명확함. M:N (학생-과목-수강)
- Capstone 프로젝트 관리: 학생·팀·과제·심사·멘토링, 일정/평가 및 역할 권한, M:N (팀-학생 + 팀-멘토)
- Conference 논문 심사: 제약(중복 or COI: Conflict of Interest) 관련 trigger, M:N (논문-저자 + 논문-심사자)
- 스포츠센터 수강·출결 관리: 회원·강사·강좌 (M:N 조건 및 8개 이상의 Entity 조건 만족)
- 병원 외래·검사 예약/보험 청구: 복잡한 실세계 규칙 및 마스킹·RBAC·Audit log 등 보안/개인정보 기능 활용

Project Guideline

■ 프로젝트 주제 선정 및 평가 기준 (가중치)

기준	질문	가중치	예시
학습 목표 적합성	정규화, 무결성, 트랜잭션, 인덱스, 뷰, 프로시저, 트리거 등 학습 내용 목표 달성	20%	최소 5개 이상 명시
데이터 모델 복잡도	Entity 8~13개 포함, M:N 포함, weak entity 포함, 상속 등	10%	RDB 설계의 타당성
시나리오	동시성/경합이 발생하는 실제 흐름이 있나?	10%	최소 1개 이상의 규칙
무결성/제약	PK/FK, 유니크, 체크, 부분적 NULL 전략	10%	무결성/제약 설계가 명확한지
질의 난이도	CTE, Join, Subquery, 집계, 롤업 등	10%	중/고급 SQL 5건 이상
성능/튜닝	인덱스 설계, 실행계획 비교 등	10%	전/후 성능 비교 포함
보안/개인정보	Masking, Audit log, RBAC 등	10%	2개 이상 구현 여부
UI 및 동작	Web application (UI 포함) 동작 여부	20%	페이지 동작 여부

Project Guideline

- 프로젝트 주제 선정 (스포츠센터)

- 기본 요구사항

1. 회원 등록(연령대/지역/연락처), 연락처 마스킹 조회
2. 강좌(수영/요가/헬스) 개설: 요일/시간/정원/수강료/수강기간
3. 수강신청(회원×강좌), 정원 초과·중복시간 차단 규칙
4. 출결(날짜별 출석/지각/결석) 기록 및 출석률 계산 뷰
5. 강사 배정(한 강사 여러 강좌), 강사 급여 산출(시급×실강의시간)
6. 환불 규정(개강 전/후 차등), 환불 요청/승인 처리
7. 시설 예약(수영장 레인/다목적실)과 중복 예약 방지
8. 공지/이벤트, 만족도 설문
9. 간단 RBAC(관리자/강사/회원) 및 권한 별 화면
10. 보고서: 강좌별 등록/이탈률, 연령대별 선호, 시간대별 이용률

Project Guideline

■ 프로젝트 주제 선정 (병원)

➤ 기본 요구사항

1. 환자 등록/수정(실명·연락처·생년·보험유형), 민감정보 마스킹 보기(예: 전화 뒷자리) 제공
2. 의사·진료과·진료실 관리 및 근무 스케줄(슬롯) 등록
3. 외래 예약(환자×의사×시간) 생성/변경/취소, 이중예약 방지 규칙
4. 검사실(CT/MRI/채혈 등)과 장비 단위 슬롯 예약, 준비절차(금식 등) 체크
5. 진료 후 처방전/검사 처방 발행, 약국/검사실 전송 규 관리
6. 수납/결제(진료·검사 비용), 할인/보험 본인부담 계산 로직
7. 보험 청구(요양기호, 급여/비급여 구분, 청구상태) 및 반려 사유 기록
8. No-show/지각 정책과 패널티/알림 기록
9. 환자용 포털: 예약 조회/변경, 검사 전 준수사항 확인, 안내문 다운로드
10. 의사용 포털: 당일 리스트/차트 보기, 처방 입력, 예약 변경 승인
11. 감사 로그: 환자 차트 열람/수정/다운로드 이력, 관리자 계정 활동
12. 보고서: 진료과별 환자수·수익, 장비 가동률, 보험 반려율

Project Guideline

- 보고서 작성 가이드라인

- 학습 목표 정합성 (20%)

- 정규화: 1NF/2NF/3NF 적용 근거(함수적 종속, 이상현상 제거 사례)
 - 트랜잭션 경계: "예약 확정", "결제 확정", "재고 차감" 등 단위 정의와 **ACID 목표**
 - 인덱스: PK/UK/보조 인덱스 설계 사유(카디널리티, 접근패턴, 커버링 여부)
 - 뷰: 보안/편의 목적의 뷰 2~3개(예: 민감정보 마스킹 뷰, 통계 집계 뷰)
 - 프로시저/함수: 업무 규칙 캡슐화(예: 수강료 계산, 환불 계산)
 - 트리거: 무결성 보조(예: 이중 예약 방지, 감사 로그 자동 기록)

Project Guideline

- 보고서 작성 가이드라인

➢ 정규화: 1NF/2NF/3NF 적용 근거(Atomic 사례/함수적 종속 제거 사례/이행적 종속 제거 사례/이상현상 제거 사례)

주문번호	고객이름	상품목록
1	KIM	TV, Cellphone
2	RHO	Microwave
3	LEE	Laptop, Monitor

1NF 문제

학번	과목코드	학생이름	과목명	성적
2025001	IS101	KIM	Cybersecurity	A
2025001	MA101	KIM	Calculus	B
2025002	CS201	RHO	C Programming	C

2NF 문제 (stu_grade)

학번	이름	학과번호	학과명
2025001	KIM	101	Industrial Security
2025002	RHO	102	Computer Engineering
2025003	LEE	101	Industrial Security

3NF 문제

Project Guideline

- 보고서 작성 가이드라인

- **트랜잭션 경계**: "예약 확정", "결제 확정", "재고 차감" 등 단위 정의와 **ACID**(Atomicity, Consistency, Isolation, Durability) 목표

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;  
START TRANSACTION;
```

....

- 1) 주문 상태 확인
- 2) 재고가 충분한지 체크
- 3) 재고 차감 후 원장 기록
- 4) 주문완료 후 결제 상태로 이동

.....

```
COMMIT; 또는 ROLLBACK;
```

* (데이터베이스) 트랜잭션: 데이터베이스의 상태를 변화시키는 프로그램의 작업 단위

Project Guideline

* 카디널리티(Cardinality): 컬럼의 고유한 값의 개수를 나타내는 수치

■ 보고서 작성 가이드라인

➤ 인덱스: PK/UK/보조 인덱스 설계 사유(카디널리티, 접근패턴, 커버링 여부)

```
mysql> EXPLAIN ANALYZE
-> SELECT
->   d.dept_name, e.emp_no, e.first_name, e.last_name,
->   s.salary, t.title
-> FROM employees e
-> JOIN dept_emp   de ON de.emp_no = e.emp_no
-> JOIN departments d ON d.dept_no = de.dept_no
-> JOIN salaries   s ON s.emp_no = e.emp_no
-> JOIN titles     t ON t.emp_no = e.emp_no
-> WHERE
->   de.to_date = '9999-01-01'
->   AND s.to_date = '9999-01-01'
->   AND t.to_date = '9999-01-01'
->   AND e.hire_date BETWEEN '1990-01-01' AND '1995-12-31'
->   AND s.salary BETWEEN 80000 AND 90000
->   AND d.dept_name IN ('Sales','Development');
```



```
mysql> ALTER TABLE dept_emp ADD INDEX idx_de_current (to_date, dept_no, emp_no);
Query OK, 0 rows affected (10.78 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE salaries ADD INDEX idx_sal_current (to_date, salary, emp_no);
Query OK, 0 rows affected (1 min 6.22 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
-> Nested loop inner join (cost=30.00 rows=2) (actual time=16.709..3159.833 rows=5128 loops=1)
-> Nested loop inner join (cost=16.82 rows=7) (actual time=0.060..1478.649 rows=37501 loops=1)
-> Nested loop inner join (cost=11.25 rows=9) (actual time=0.052..693.805 rows=37501 loops=1)
-> Nested loop inner join (cost=4.64 rows=19) (actual time=0.033..125.860 rows=99087 loops=1)
-> Filter: (d.dept_name in ('Sales','Development')) (cost=0.67 rows=2) (actual time=0.020..0.045 rows=2 loops=1)
-> Covering index range scan on d using idx_dept_name over (dept_name = 'Development') OR (dept_name = 'Sales')
-> Filter: (e.hire_date between '1990-01-01' and '1995-12-31') (cost=0.25 rows=0) (actual time=0.006..0.006 rows=0 loops=1)
-> Single-row index lookup on e using PRIMARY (emp_no=de.emp_no) (cost=0.25 rows=1) (actual time=0.005..0.005 rows=1 loops=1)
-> Filter: (t.to_date = DATE'9999-01-01') (cost=0.45 rows=1) (actual time=0.020..0.021 rows=1 loops=37501)
-> Index lookup on t using PRIMARY (emp_no=de.emp_no) (cost=0.45 rows=1) (actual time=0.020..0.021 rows=1 loops=37501)
-> Filter: ((s.to_date = DATE'9999-01-01') and (s.salary between 80000 and 90000)) (cost=0.94 rows=0) (actual time=0.045..0.045 rows=0 loops=37501)
-> Index lookup on s using PRIMARY (emp_no=de.emp_no) (cost=0.94 rows=10) (actual time=0.043..0.044 rows=9 loops=37501)
```



```
-> Nested loop inner join (cost=16921.43 rows=14) (actual time=125.689..4245.734 rows=5128 loops=1)
-> Nested loop inner join (cost=16654.73 rows=135) (actual time=69.584..2390.052 rows=37501 loops=1)
-> Nested loop inner join (cost=15599.51 rows=920) (actual time=68.242..1713.182 rows=37501 loops=1)
-> Nested loop inner join (cost=12702.01 rows=8279) (actual time=68.213..1094.823 rows=99087 loops=1)
-> Filter: (d.dept_name in ('Sales','Development')) (cost=1.43 rows=2) (actual time=51.567..72.763 rows=2 loops=1)
-> Covering index range scan on d using dept_name over (dept_name = 'Development') OR (dept_name = 'Sales') (cost=1.43 rows=2) (actual time=51.563..72.756 rows=2 loops=1)
-> Filter: (de.to_date = DATE'9999-01-01') (cost=2417.96 rows=4139) (actual time=18.991..508.604 rows=49544 loops=2)
-> Index lookup on de using dept_no (dept_no=d.dept_no) (cost=2417.96 rows=4139) (actual time=18.988..503.442 rows=68976 loops=2)
-> Filter: (e.hire_date between '1990-01-01' and '1995-12-31') (cost=0.25 rows=0) (actual time=0.006..0.006 rows=0 loops=99087)
-> Single-row index lookup on e using PRIMARY (emp_no=de.emp_no) (cost=0.25 rows=1) (actual time=0.006..0.006 rows=1 loops=99087)
-> Filter: (t.to_date = DATE'9999-01-01') (cost=1.00 rows=0) (actual time=0.017..0.018 rows=1 loops=37501)
-> Index lookup on t using PRIMARY (emp_no=de.emp_no) (cost=1.00 rows=1) (actual time=0.017..0.018 rows=1 loops=37501)
-> Filter: ((s.to_date = DATE'9999-01-01') and (s.salary between 80000 and 90000)) (cost=1.01 rows=0) (actual time=0.049..0.049 rows=0 loops=37501)
-> Index lookup on s using PRIMARY (emp_no=de.emp_no) (cost=1.01 rows=10) (actual time=0.047..0.049 rows=9 loops=37501)
```

Project Guideline

- 보고서 작성 가이드라인

- 데이터 모델 복잡도 (10%) — "Entity 8~13개, M:N, Weak entity, Inheritance"

- 엔티티 수: 핵심 8~13개 표로 나열(이름, 핵심 속성, PK, FK, UK, 예상 행 수)
 - 관계: 1:N, M:N(교차 테이블), 약한 엔티티(예: employee(강) <- dependent(약), Flight <- seat), 상속(예: 사람 → 직원/회원 또는 메뉴 → 세트/옵션)
 - 도식: 최종 EER 다이어그램(ERD 스냅샷 포함) + 명세표

Project Guideline

- 보고서 작성 가이드라인

- 시나리오(동시성/경합) (10%) — “최소 1개 이상의 규칙”

- 핵심 시나리오 흐름도: 예) “사용자 A와 B가 같은 슬롯(장비, 강의실 등)을 동시에 예약”
 - 락/격리수준 제안: READ COMMITTED/REPEATABLE READ/Serializable 중 선택 사유
 - 경합 규칙: “정원 초과 불가”, “재고 음수 불가”
 - 실험: 동시 트랜잭션 시나리오 재현 스크립트(2개 세션 용 SQL)와 결과

Project Guideline

- 보고서 작성 가이드라인

- 무결성/제약 (10%)

- PK/FK/UNIQUE/NOT NULL/CHECK 명세표(컬럼, 제약명, 이유)
 - 부분 NULL 전략: 언제 허용/불허(예: 보험 미가입 NULL 허용)
 - 참조 무결성: ON DELETE/UPDATE 규칙(Restrict/Cascade/Set Null) 근거

Project Guideline

- 보고서 작성 가이드라인

- 질의 난이도 (10%) — “중/고급 SQL 5건 이상”

- **복합 JOIN**: 환자×진료×검사×결제, 주문×재고 소진내역
 - **서브쿼리/EXISTS**: 최근 90일 내 반복 반려 청구 건
 - **집계/롤업**: 시간대별 매출, 과별 환자수 ROLLUP
 - **권한/마스킹 뷰 연계 쿼리**: RBAC 등급에 따라 마스킹 적용 결과 비교

Project Guideline

■ 보고서 작성 가이드라인

➤ 성능/튜닝 (10%) — “전/후 비교 포함”

- 핵심 쿼리 3~5건 선택 → EXPLAIN 계획/실행시간 측정(행수·필터율 포함)
- 인덱스 or 파티션 or 커버링 인덱스 적용 전후 비교 표
- 안티패턴 교정: N+1 쿼리, 함수 기반 인덱스 부재, 와일드카드 선두 like 등

```
SELECT * FROM orders WHERE YEAR(order_date) = 2025;
```

```
ALTER TABLE orders ADD COLUMN order_year INT AS (YEAR(order_date));  
CREATE INDEX idx_order_year ON orders (order_year);
```

```
→ SELECT * FROM orders WHERE order_year = 2025;
```

```
SELECT * FROM employees WHERE first_name LIKE '%tom%';
```

```
→ SELECT * FROM employees WHERE first_name LIKE 'tom%';
```

```
ALTER TABLE employees ADD FULLTEXT INDEX ft_firstname (first_name);
```

```
→ SELECT * FROM employees WHERE MATCH(first_name) AGAINST('tom' IN BOOLEAN MODE);
```

Project Guideline

■ 보고서 작성 가이드라인

➤ 보안/개인정보 (10%) — "2개 이상 구현"

- **마스킹**: 전화/주민번호/카드번호 마스킹 뷰 + 비마스킹 권한 제한
- **RBAC**: 역할(예: admin/doctor/nurse/patient, manager/staff, admin/instructor) 정의, 역할별 CRUD 매트릭스
- **감사 로그**: 읽기/쓰기/권한 변경 기록(누가/언제/무엇을/어디서)

```
INSERT INTO employees VALUES (1, 'KIM', '010-1234-5678', '020202-3234567', 'Student')
CREATE VIEW v_user_masked AS
SELECT user_id, name, CONCAT(LEFT(phone, 4), '****-****') AS phone,
      CONCAT(LEFT(ssn, 8), '*****') AS ssn                                FROM users;

CREATE VIEW v_user_full AS SELECT * FROM users WHERE role='admin';
```

역할 (Role)	Create	Read	Update	Delete
Admin	O	O	O	O
Professor	O	O	O	X
Staff	X	O	X	X
Student	X	O (본인만)	X	X

Project Guideline

- 보고서 작성 가이드라인

- UI 및 동작 (20%)

- 최소 동작 페이지 목록: 목록/상세/생성/수정/취소/통계(역할별)
 - 예외 처리: 중복 예약/정원 초과/재고 부족/결제 실패 플로우
 - 데모 스크립트: 시나리오별 클릭 순서 + 확인할 DB 결과(캡처/쿼리)

Project Guideline

- 보고서 작성 가이드라인

- 제출물

- ERD(EER), { DDL 스크립트, 원본데이터, 뷰/인덱스/제약/트리거/프로시저 코드 } → 백업(SQL) 파일로 제출
 - 테스트 시나리오 SQL(동시성 1건 포함)
 - 보안/개인정보 설계서, RBAC 표, 감사 로그 테이블 샘플
 - 성능 보고서(튜닝 전후 EXPLAIN/시간 비교)
 - 최종 데모용 체크리스트(페이지 및 평가기준에 해당하는 항목 동작 여부) 및 결과(화면) 캡처