

3과목

CHAPTER 01 SQL 응용

048 데이터베이스와 SQL★★★

• 절차형 SQL 종류

종류	설명
트리거 (Trigger)	- 테이블에 대한 이벤트에 반응해 자동으로 실행되는 작업 - 데이터 작업 제한과 기록, 변경 작업 감사 등을 수행
프로시저 (Procedure)	- 어떤 행동을 수행하기 위한 일련의 작업 순서 - 일련의 연산 처리 결과를 마치 하나의 함수처럼 실행
사용자 정의 함수 (User-Defined Function)	절차형 SQL을 사용하여 일련의 SQL 처리를 단일 값으로 반환할 수 있도록 수행

• SQL 문법의 종류

종류	설명	명령어	역할
데이터 정의어 DDL (Data Definition Language)	- 데이터베이스를 정의하는 언어 - 데이터의 전체적인 구조를 결정하는 역할	CREATE	테이블 생성
		ALTER	테이블 수정
		DROP	테이블 삭제
		TRUNCATE	테이블 초기화
데이터 조작어 DML (Data Manipulation Language)	데이터베이스에 저장된 자료를 조회, 수정, 삭제 등의 역할을 하는 언어	SELECT	데이터 조회
		INSERT	데이터 삽입
		UPDATE	데이터 수정
		DELETE	데이터 삭제
데이터 제어어 DCL (Data Control Language)	데이터 보안, 무결성, 회복, 병행 수행제어 등을 정의하는 언어	GRANT	작업 권한 부여
		REVOKE	작업 권한 회수
		COMMIT	작업 완료
		ROLLBACK	작업 취소, 복구

• 스키마

	외부 스키마 (External Schema)	- 사용자나 개발자의 관점에서 필요로 하는 논리적인 구조를 정의 - 하나의 데이터베이스 시스템에서 여러 외부 스키마가 존재
	개념 스키마 (Conceptual Schema)	- 데이터베이스의 논리적인 구조 - 데이터를 종합한 조직 전체의 데이터베이스로 하나만 존재 - 객체 간의 관계와 제약조건, 접근 권한, 보안 및 무결성에 관한 명세를 정의
	내부 스키마 (Internal Schema)	- 물리적 저장장치의 관점에서 본 데이터베이스 구조 - 저장 데이터 항목의 표현 방법과 내부 레코드의 물리적 순서 등을 나타냄

049 DDL★★★

• 테이블 생성 문법

문법	설명	
CREATE TABLE 테이블명 (속성명 데이터_타입 [DEFAULT 기본값] [NOT NULL], ... , PRIMARY KEY(기본키) , UNIQUE(속성명) , FOREIGN KEY(외래키) REFERENCES 참조테이블(기본키) , CONSTRAINT 제약조건명 CHECK(조건식));	PRIMARY KEY	- 기본키로 사용할 속성의 집합을 정의 - 레코드를 구별할 수 있는 역할
	FOREIGN KEY	외래키로 테이블 내의 열 중 다른 테이블을 참조하는 열
	UNIQUE	테이블 내 유일해야 하는 값으로 중복을 허용하지 않음
	CONSTRAINT	- 무결성 제약조건을 정의하며, 데이터 오류 발생 가능성을 줄임 - 제약 조건의 이름
	CHECK	제약 조건을 정의
	DEFAULT	기본값을 설정

• 테이블 정의 문법

문법	설명	예제
ALTER TABLE 테이블명 ADD 속성 데이터타입	- 테이블에 속성을 추가 - DEFAULT 정의 가능	ALTER TABLE 고객 ADD 이름 VARCHAR(9) DEFAULT '없음'
ALTER TABLE 테이블명 MODIFY 속성 데이터타입	- 테이블에 속성을 변경 - DEFAULT, NOT NULL 등 제약조건 변경 가능	ALTER TABLE 고객 MODIFY 이름 VARCHAR(20) NOT NULL
ALTER TABLE 테이블명 DROP 속성	테이블 속성을 삭제	ALTER TABLE 고객 DROP 이름
ALTER TABLE 테이블명 RENAME COLUMN 속성 TO 변경할_속성명	테이블의 속성명을 변경	ALTER TABLE 고객 RENAME COLUMN 이름 TO 고객이름

• 테이블 삭제 문법

문법	설명	예제
DROP TABLE 테이블명 [CASCADE RESTRICT]	- 테이블 제거 - CASCADE는 참조하는 다른 모든 개체를 제거 - RESTRICT 다른 테이블이 삭제할 테이블을 참조 중이면 제거하지 않음	DROP TABLE 고객 CASCADE
TRUNCATE TABLE 테이블명	- 테이블을 초기 상태로 변경 - DROP과 달리 테이블 구조만 남기고 데이터를 다 삭제함 - Rollback 불가능 - DELETE에 비해 빠르게 데이터를 제거하는 것이 가능	TRUNCATE TABLE 고객

050 DCL★★★

• GRANT와 REVOKE

GRANT		
문법	설명	
GRANT 권한 ON 테이블 TO 사용자 [WITH GRANT OPTION]	- 사용자에게 권한을 부여 - 권한 종류	
	권한	허용 내용
	ALL	모든 권한
	SELECT, INSERT, UPDATE, DELETE	레코드 조회, 입력, 수정, 삭제
	CREATE, ALTER, DROP	테이블 생성, 변경, 삭제
	- WITH GRANT OPTION 다른 사용자에게 권한을 부여할 수 있는 권한을 부여	
예시		
GRANT SELECT ON 고객 TO 매니저 WITH GRANT OPTION	'고객' 테이블에 대한 'SELECT' 권한을 '매니저'에게 부여 다른 사용자에게도 부여할 수 있는 권한 부여	
REVOKE		
명령어	설명	
REVOKE [GRANT OPTION FOR] ON 테이블 FROM 사용자 [CASCADE CONSTRAINTS]	- 사용자에게 권한을 회수 - 권한 종류는 GRANT와 같음 - CASCADE CONSTRAINTS 권한 회수 시 권한을 부여받았던 사용자가 부여한 권한도 회수	
예시		
REVOKE SELECT ON 고객 FROM 매니저	'매니저'에게 부여한 '고객' 테이블의 'SELECT' 권한 회수	

• COMMIT, ROLLBACK 그리고 SAVEPOINT

명령어	설명
COMMIT	트랜잭션이 성공적으로 끝나면 데이터베이스가 일관성 있는 상태를 가지기 위해 변경 사항을 영구적으로 적용
ROLLBACK	하나의 트랜잭션이 비정상적으로 종료 시 보류 중인 모든 변경 사항을 폐기하고 이전 상태로 회귀
SAVEPOINT	Rollback 할 포인트 지정

051 DML★★★

• SELECT

구문	설명	내용
SELECT [ALL DISTINCT] [테이블.]속성명 [AS 별칭] [,] FROM 테이블명. [, ...] [WHERE 조건] [GROUP BY 속성명, ...] [HAVING 조건] [ORDER BY 속성명[, ...] [ASC DESC]]	SELECT (필수)	- All: 모든 데이터를 반환 - DISTINCT: 중복 튜플 발견 시 그 중 첫 번째 하나만 검색 - 검색하고자 하는 속성명 - AS를 사용해 별칭으로 표시
	FROM (필수)	- 열 참조를 가진 테이블을 지정 - 테이블 명도 별칭으로 저장 가능
	WHERE	- 데이터를 추출하는 선택 조건식을 지정 - 테이블 간의 결합할 때 그 결합 관계를 지정
	GROUP BY	그룹화할 열 또는 속성명을 지정

	HAVING	GROUP BY 절에 집계한 결과에 조건을 정함
	ORDER BY	정렬한 속성명을 지정
	ASC DESC	- ASC: Asending Order 오름차순(기본값) - DESC: Descending Order 내림차순

• SELECT 자세한 설명

구문	설명																
SELECT [테이블명.]속성명 [AS 별칭] [. ...]	- 검색하고자 하는 속성명 - AS를 사용해 별칭으로 표시 가능 - 여러 속성을 선택 가능																
[분석함수(속성) OVER ([PARTITION BY 속성명 [. ...]])]	- PARTITION BY로 구분된 집합을 WINDOW라고 함 - PARTITION BY에 지정한 속성이 WINDOW 범위임 - GROUP BY를 사용하지 않고 속성의 값을 집계함																
집계 함수	- 집계 함수 여러 행 또는 테이블 전체 행으로부터 하나의 집계 결과를 반환 <table border="1"> <thead> <tr> <th>집계 함수</th><th>설명</th></tr> </thead> <tbody> <tr> <td>COUNT</td><td>속성별 범위의 튜플의 수를 집계</td></tr> <tr> <td>SUM</td><td>속성 범위의 합계를 집계</td></tr> <tr> <td>AVG</td><td>속성 범위의 평균을 집계</td></tr> <tr> <td>MAX</td><td>속성 범위의 최댓값을 집계</td></tr> <tr> <td>MIN</td><td>속성 범위의 최솟값을 집계</td></tr> <tr> <td>STDDEV</td><td>속성 범위의 표준편차를 집계</td></tr> <tr> <td>VARIANCE</td><td>속성 범위의 분산을 집계</td></tr> </tbody> </table>	집계 함수	설명	COUNT	속성별 범위의 튜플의 수를 집계	SUM	속성 범위의 합계를 집계	AVG	속성 범위의 평균을 집계	MAX	속성 범위의 최댓값을 집계	MIN	속성 범위의 최솟값을 집계	STDDEV	속성 범위의 표준편차를 집계	VARIANCE	속성 범위의 분산을 집계
집계 함수	설명																
COUNT	속성별 범위의 튜플의 수를 집계																
SUM	속성 범위의 합계를 집계																
AVG	속성 범위의 평균을 집계																
MAX	속성 범위의 최댓값을 집계																
MIN	속성 범위의 최솟값을 집계																
STDDEV	속성 범위의 표준편차를 집계																
VARIANCE	속성 범위의 분산을 집계																
[RANK, ROW_NUMBER] DENSE_RANK,	- 순위함수 파티션에서 각 행의 순위 값을 반환 <table border="1"> <thead> <tr> <th>순위 함수</th><th>설명</th></tr> </thead> <tbody> <tr> <td>RANK</td><td>- 중복 순위 시 다음 값은 넘어감 - 예시 데이터: 10, 10, 5 - 순서: 1, 1, 3</td></tr> <tr> <td>ROW_NUMBER</td><td>- 중복 순위 시 순차적인 순위를 표시 - 예시 데이터: 10, 10, 5 - 순서: 1, 2, 3</td></tr> <tr> <td>DENSE_RANK</td><td>- 중복 순위 시 순차적 순서 부여 - 예시 데이터: 10, 10, 5 - 순서: 1, 1, 2</td></tr> </tbody> </table>	순위 함수	설명	RANK	- 중복 순위 시 다음 값은 넘어감 - 예시 데이터: 10, 10, 5 - 순서: 1, 1, 3	ROW_NUMBER	- 중복 순위 시 순차적인 순위를 표시 - 예시 데이터: 10, 10, 5 - 순서: 1, 2, 3	DENSE_RANK	- 중복 순위 시 순차적 순서 부여 - 예시 데이터: 10, 10, 5 - 순서: 1, 1, 2								
순위 함수	설명																
RANK	- 중복 순위 시 다음 값은 넘어감 - 예시 데이터: 10, 10, 5 - 순서: 1, 1, 3																
ROW_NUMBER	- 중복 순위 시 순차적인 순위를 표시 - 예시 데이터: 10, 10, 5 - 순서: 1, 2, 3																
DENSE_RANK	- 중복 순위 시 순차적 순서 부여 - 예시 데이터: 10, 10, 5 - 순서: 1, 1, 2																

• GROUP BY, ORDER BY 자세한 설명

구문	설명
GROUP BY 속성명[. ...]	정의한 속성값을 기준으로 그룹화하여 결과를 처리
[ROLLUP(속성명, [. ...])]	데이터를 그룹화할 기준 열을 지정
[CUBE(속성명, [. ...])]	ROLLUP과 유사하나 결합 가능한 모든 값의 소계를 구함
[HAVING 조건]	- GROUP BY가 있는 경우 사용 가능 - 그룹에 대한 조건을 지정
[ORDER BY 속성명, [. ...]]	속성명을 대상으로 정렬
[ASC DESC]	속성명 정렬 시 오름차순(ASC), 내림차순(DESC)을 정의하고, 지정하지 않은 경우 기본값으로 오름차순 정렬

• WHERE 비교 연산자

연산자	의미	
관계	A = B	A와 B가 같다
	A <> B	A와 B가 같지 않다
	A < B	A보다 B가 크다
	A > B	A가 B보다 크다
	A >= B	A가 B보다 같거나 크다
	A <= B	B가 A보다 같거나 크다
논리	NOT A	A가 TRUE가 아닌 경우 TRUE를 반환
	A AND B	A와 B 둘 다 TRUE인 경우 TRUE를 반환
	A OR B	A혹은 B중 하나만 TRUE여도 TRUE를 반환
패턴	A 속성 입력 예시: '333', 'ABC', '3AB', '33', '3K'	
	A LIKE '3%'	%: 모든 문자를 검색 3으로 시작하는 모든 문자('333', '3AB', '33', '3K')
		A LIKE '3_'
	A LIKE '3#'	
		BETWEEN
IN	- 속성이 특정 값을 가지고 있는 값을 검색 - A IN(10, 20, 30): A 속성값이 10, 20, 30인 경우	
NULL	- 데이터값이 존재하지 않음을 나타내기 위해 사용 - A IS NULL: A 값이 존재하지 않는 경우 - A IS NOT NULL:A 값이 존재하는 경우	
산술	- 산술, 관계, 논리 연산자 순서로 우선순위를 가짐 - x, /, +, - 산술 연산으로, 왼쪽이 더 우선순위가 높음	
EXISTS	- 서브 쿼리에만 사용할 수 있다. - EXISTS: 데이터가 존재하는 경우 TRUE를 반환 - NOT EXISTS: 데이터가 존재하는 않는 경우 TRUE를 반환	

• UPDATE

구문	설명
UPDATE 테이블명	수정하고자 하는 대상 "테이블 명"을 지정
SET 속성명 = 데이터 [, ...]	- 수정하고자 하는 속성명과 값을 지정 - "속성명"의 속성을 "데이터"로 변경
[WHERE 조건]	- 수정할 레코드를 선택할 조건을 지정

• DELETE

구문	설명
DELETE FROM 테이블명	"테이블명"의 테이블 명에서 행을 삭제
[WHERE 조건]	- 삭제 조건을 설정 - WHERE 조건 없이 수행 시 모든 레코드 삭제

• INSERT

구문	설명
INSERT INTO 테이블	데이터를 삽입할 테이블을 정의
(속성명,)	데이터를 삽입할 속성을 정의
VALUES(데이터,)	- 데이터를 정의 - 속성, 데이터 수, 타입이 일치해야 함

052 집합연산자★

• 집합 연산자

집합 연산자	설명									
UNION	중복을 제거하여 새로 집합을 생성함									
	<table><tr><th>구문</th><th>결과</th></tr><tr><td>(SELECT 학번 FROM R1) UNION (SELECT 학번 FROM R2)</td><td><table><tr><th>학번</th></tr><tr><td>20201111</td></tr><tr><td>20202222</td></tr><tr><td>20203333</td></tr></table></td></tr></table>	구문	결과	(SELECT 학번 FROM R1) UNION (SELECT 학번 FROM R2)	<table><tr><th>학번</th></tr><tr><td>20201111</td></tr><tr><td>20202222</td></tr><tr><td>20203333</td></tr></table>	학번	20201111	20202222	20203333	
	구문	결과								
(SELECT 학번 FROM R1) UNION (SELECT 학번 FROM R2)	<table><tr><th>학번</th></tr><tr><td>20201111</td></tr><tr><td>20202222</td></tr><tr><td>20203333</td></tr></table>	학번	20201111	20202222	20203333					
학번										
20201111										
20202222										
20203333										
UNION ALL	중복된 행을 모두 유지하면서 행을 합침									
	<table><tr><th>구문</th><th>결과</th></tr><tr><td>(SELECT 학번 FROM R1) UNION ALL (SELECT 학번 FROM R2)</td><td><table><tr><th>학번</th></tr><tr><td>20201111</td></tr><tr><td>20202222</td></tr><tr><td>20202222</td></tr><tr><td>20203333</td></tr></table></td></tr></table>	구문	결과	(SELECT 학번 FROM R1) UNION ALL (SELECT 학번 FROM R2)	<table><tr><th>학번</th></tr><tr><td>20201111</td></tr><tr><td>20202222</td></tr><tr><td>20202222</td></tr><tr><td>20203333</td></tr></table>	학번	20201111	20202222	20202222	20203333
	구문	결과								
(SELECT 학번 FROM R1) UNION ALL (SELECT 학번 FROM R2)	<table><tr><th>학번</th></tr><tr><td>20201111</td></tr><tr><td>20202222</td></tr><tr><td>20202222</td></tr><tr><td>20203333</td></tr></table>	학번	20201111	20202222	20202222	20203333				
학번										
20201111										
20202222										
20202222										
20203333										
INTERSECT	공통된 행만 추출하여 새로운 집합을 생성함									
	<table><tr><th>구문</th><th>결과</th></tr><tr><td>(SELECT 학번 FROM R1) INTERSECT (SELECT 학번 FROM R2)</td><td><table><tr><th>학번</th></tr><tr><td>20202222</td></tr></table></td></tr></table>	구문	결과	(SELECT 학번 FROM R1) INTERSECT (SELECT 학번 FROM R2)	<table><tr><th>학번</th></tr><tr><td>20202222</td></tr></table>	학번	20202222			
	구문	결과								
(SELECT 학번 FROM R1) INTERSECT (SELECT 학번 FROM R2)	<table><tr><th>학번</th></tr><tr><td>20202222</td></tr></table>	학번	20202222							
학번										
20202222										
MINUS, EXCEPT	첫 번째 테이블에서 두 번째 테이블의 결과를 제외한 나머지 결과를 반환함									
	<table><tr><th>구문</th><th>결과</th></tr><tr><td>(SELECT 학번 FROM R1) MINUS (SELECT 학번 FROM R2)</td><td><table><tr><th>학번</th></tr><tr><td>20201111</td></tr></table></td></tr></table>	구문	결과	(SELECT 학번 FROM R1) MINUS (SELECT 학번 FROM R2)	<table><tr><th>학번</th></tr><tr><td>20201111</td></tr></table>	학번	20201111			
	구문	결과								
(SELECT 학번 FROM R1) MINUS (SELECT 학번 FROM R2)	<table><tr><th>학번</th></tr><tr><td>20201111</td></tr></table>	학번	20201111							
학번										
20201111										

053 조인(JOIN)★

• JOIN 구문

JOIN 구문	설명								
INNER JOIN	<p> - 두 개 이상의 테이블에서 공통된 값을 기준으로 연결하여 새로운 결과 집합을 생성 - 교집합의 개념 - 고객 테이블과 주문 테이블을 고객 ID로 JOIN </p> <table> <tr> <td>문법 1</td><td> SELECT [테이블명.]속성명 [, ...] FROM 테이블명1 [INNER] JOIN 테이블2 ON 조인조건 [WHERE 조건] </td></tr> <tr> <td>예시 1</td><td> SELECT * FROM 고객 INNER JOIN 주문 ON 고객.고객ID = 주문.고객ID; </td></tr> <tr> <td>문법 2</td><td> SELECT [테이블명.]속성명 [, ...] FROM 테이블명1, 테이블명2, ... WHERE 테이블명1.속성명 = 테이블명2.속성명 </td></tr> <tr> <td>예시 2</td><td> SELECT * FROM 고객, 주문 WHERE 고객.고객ID = 주문.고객ID; </td></tr> </table>	문법 1	SELECT [테이블명.]속성명 [, ...] FROM 테이블명1 [INNER] JOIN 테이블2 ON 조인조건 [WHERE 조건]	예시 1	SELECT * FROM 고객 INNER JOIN 주문 ON 고객.고객ID = 주문.고객ID;	문법 2	SELECT [테이블명.]속성명 [, ...] FROM 테이블명1, 테이블명2, ... WHERE 테이블명1.속성명 = 테이블명2.속성명	예시 2	SELECT * FROM 고객, 주문 WHERE 고객.고객ID = 주문.고객ID;
문법 1	SELECT [테이블명.]속성명 [, ...] FROM 테이블명1 [INNER] JOIN 테이블2 ON 조인조건 [WHERE 조건]								
예시 1	SELECT * FROM 고객 INNER JOIN 주문 ON 고객.고객ID = 주문.고객ID;								
문법 2	SELECT [테이블명.]속성명 [, ...] FROM 테이블명1, 테이블명2, ... WHERE 테이블명1.속성명 = 테이블명2.속성명								
예시 2	SELECT * FROM 고객, 주문 WHERE 고객.고객ID = 주문.고객ID;								

OUTER JOIN	<ul style="list-style-type: none"> - 두 개 이상의 테이블에서 공통된 값을 기준으로 하나의 테이블에서만 존재하는 데이터도 결과 집합에 포함 - 일치하지 않는 경우 NULL 값을 가짐 	
	OUTER JOIN 종류	설명
	LEFT OUTER JOIN	왼쪽 테이블의 모든 레코드와 오른쪽 테이블에서 일치하는 레코드만을 출력
	RIGHT OUTER JOIN	오른쪽 테이블의 모든 레코드와 왼쪽 테이블에서 일치하는 레코드만을 출력
	FULL OUTER JOIN	왼쪽과 오른쪽 테이블에서 LEFT OUTER JOIN 후 RIGHT OUTER JOIN을 둘 다 실행
	문법 1	SELECT [테이블명.]속성명 [, ...] FROM 테이블명1, 테이블명2, ... WHERE 테이블명1.속성명[(+)] = 테이블명2.속성명[(+)]
	문법 2	SELECT [테이블명.]속성명 [, ...] FROM 테이블1(LEFT RIGHT FULL) [OUTER] JOIN 테이블2 [WHERE 조건]
SELF JOIN	<ul style="list-style-type: none"> - 하나의 테이블에서 자신과 조인하는 방법 - 서로 다른 레코드 간의 관계를 분석하거나 계층 구조를 가진 데이터를 조회하는 경우에 사용 	
	문법 1	SELECT [테이블명.]속성명 [, ...] FROM 테이블명1 [AS] 별칭1, 테이블명1 [AS] 별칭2 [, ...] WHERE 별칭1.속성명 = 별칭2.속성명
	문법 2	SELECT [테이블명.]속성명 [, ...] FROM 테이블명1 [AS] 별칭1 JOIN 테이블명1 [AS] 별칭2 [, ...] ON 별칭1.속성명 = 별칭2.속성명 [WHERE 조건]
CROSS JOIN	<ul style="list-style-type: none"> - 두 개 이상의 테이블을 조합하여 가능한 모든 조합을 생성 - 테이블 간의 관계가 없는 경우에 사용 	
	문법 1	SELECT [테이블명.]속성명 [, ...] FROM 테이블1 CROSS JOIN 테이블2
	문법 2	SELECT [테이블명.]속성명 [, ...] FROM 테이블1, 테이블2

054 서브쿼리★

• 서브쿼리

종류	설명
단일 행 서브쿼리	<ul style="list-style-type: none"> - 서브쿼리가 하나의 행만 반환하는 경우 - 단일행 비교연산자와 사용 - =, <, <=, >=, >, <>. - 예시 <div> SELECT 이름, 연봉 FROM 사원 WHERE 연봉 = (SELECT MAX(연봉) FROM 사원) </div>
다중 행 서브쿼리	<ul style="list-style-type: none"> - 서브쿼리가 여러 개의 행을 반환하는 경우 사용 - 특정 조건을 만족하는 데이터를 조회 - "IN", "NOT IN" 비교 연산자 사용 - 예시

	SELECT * FROM 주문 WHERE 고객ID IN (SELECT 고객ID FROM 고객 WHERE 지역 = '서울')
인라인 뷰	- 서브쿼리를 사용하여 새로운 가상 테이블을 생성하는 경우 사용 - FROM절 안에 사용되는 서브 쿼리 SELECT * FROM (SELECT 상품ID, 상품명, SUM(주문량) AS 판매량 FROM 주문상품 GROUP BY 상품ID, 상품명 ORDER BY 판매량 DESC) AS 판매상품

055 뷰(VIEW)★★★

• VIEW

구분	설명		
명령어	문법	예시	설명
	CREATE VIEW 뷰명[(속성명, ...)] AS(SELECT문);	CREATE VIEW 주문정보 AS(SELECT * FROM 주문 WHERE 고객ID IN (SELECT 고객ID FROM 고객 WHERE 지역 = '서울')	VIEW 생성
	DROP VIEW 뷰명	DROP VIEW 주문정보	VIEW 삭제
장점	- 보안 강화, 논리적 독립성 제공: 뷰를 통해 특정한 컬럼만을 조회하거나, 접근 권한이 없는 사용자에게 뷰를 제공함으로써 보안을 강화 - 쿼리 간소화: 복잡한 쿼리를 뷰로 간소화하여 사용 - 데이터 무결성 유지: 특정한 조건에 따른 결과만을 조회할 수 있으므로, 데이터 무결성을 유지할 수 있음		
단점	- 데이터 변경 불가능 : 뷰는 기본적으로 READ ONLY이고, 뷰에서 데이터를 수정하려면 해당 데이터를 저장하는 실제 테이블을 수정해야 함, 뷰를 변경하려면 뷰를 다시 생성해야 함 - 인덱스 불가능		

056 인덱스★★

• 인덱스의 종류

인덱스 종류	설명
트리 기반 인덱스 (T)	- B-Tree 알고리즘 활용 - 범위 검색, 정렬, 그룹화에 유용
해시 인덱스	- (Key, Value)로 데이터를 저장하는 자료구조 중 하나로 빠른 데이터 검색 시 유용함 - 데이터 접근 비용이 균일함
비트맵 인덱스	- 0과 1의 비트 값을 사용하여 인덱스를 생성 - 다중 조건을 만족하는 튜플 개수 계산이 적합
단일 인덱스	- 읽기만으로 사용되는 경우에 사용 - 하나의 컬럼으로만 구성된 인덱스
결합 인덱스	- 여러 컬럼들을 묶어 하나의 인덱스로 사용 - 조건으로 사용하는 빈도가 높은 경우 사용
클러스터드 인덱스	- 인덱스 순서대로 데이터가 저장 - 데이터 삽입, 삭제 발생 시 순서를 유지하기 위해 데이터를 재정렬 - 특정 범위 검색 시 유리

• 인덱스

문법	설명
CREATE [UNIQUE] INDEX 인덱스명 ON 테이블명(컬럼명)	- 생성하고자 하는 인덱스 명으로 인덱스를 생성 - UNIQUE는 인덱스 컬럼의 중복 값을 허용하지 않음
DROP INDEX 인덱스명	인덱스를 삭제
ALTER [UNIQUE] INDEX 인덱스명 ON 테이블명(컬럼명)	인덱스를 변경



CHAPTER 02 SQL 활용

057 트랜잭션★★★

• 트랜잭션

종류	설명
원자성 (Atomicity)	<ul style="list-style-type: none"> - 하나의 트랜잭션이 더 이상 작게 쪼갤 수 없는 최소한의 업무 단위 - 데이터베이스에 연산이 모두 반영되었는지, 아니면 전혀 반영되지 않아야 함 - 주요 기법으로 Commit과 Rollback을 사용함
일관성 (Consistency)	<ul style="list-style-type: none"> - 트랜잭션이 완료된 결과값이 일관적인 DB 상태를 유지해야 함 - 수행하고 있는 트랜잭션에서 오류가 발생하면 현재 내역을 날려버리고 롤백해야 함
고립성 (Isolation)	<ul style="list-style-type: none"> - 하나의 트랜잭션 수행 시 다른 트랜잭션이 작업이 끼어들지 못하도록 보장함 - 트랜잭션끼리 서로 간섭할 수 없음
영속성 (Durability)	<ul style="list-style-type: none"> - 트랜잭션이 정상적으로 종료된 다음에는 영구적으로 데이터베이스에 작업의 결과가 저장되어야 함 - 성공적으로 완료된 트랜잭션의 결과는 시스템이 고장 나더라도 영구적으로 반영되어야 함

058 트랜잭션 관리 기법★★★

• 병행제어

개념	설명															
회복기법 (Recovery)	<ul style="list-style-type: none">- 회복은 데이터베이스 시스템이 장애나 오류 발생 시 복구시키는 과정- 대표적인 방법으로는 로그 기반 회복 기법, 그림자 페이지 기반 회복, 체크포인트 기반 회복이 있음															
병행제어 기법	<ul style="list-style-type: none">- 병행 제어 기법으로 로킹(Locking), 타임 스탬프(Time Stamping), 다중버전 동시제어(Multiversion Concurrency Control: MVCC), 낙관적(optimistic concurrency) 수행 있음- 로킹기법은 같은 자원을 액세스하는 다중 트랜잭션 환경에서 일관성과 무결성을 유지하기 위해 트랜잭션의 순차적 진행을 보장하는 직렬화 기법- 데이터베이스, 파일, 레코드 등은 로킹 단위가 될 수 있으며, 한꺼번에 로킹할 수 있는 단위를 로킹 단위라고 함 <table><tr><td>로킹 단위</td><td>오버헤드</td><td>로크 수</td><td>병행 수준</td><td>DB 공유도</td></tr><tr><td>커짐</td><td>감소</td><td>적어짐</td><td>낮아짐</td><td>감소</td></tr><tr><td>작아짐</td><td>증가</td><td>증가함</td><td>높아짐</td><td>증가</td></tr></table> <ul style="list-style-type: none">- 타임스탬프는 트랜잭션이 읽거나 갱신한 데이터에 대해 타임스탬프를 부여해 시간에 따라 트랜잭션 작업을 수행- 낙관적 기법은 트랜잭션이 종료된 이후에 일괄적으로 검사하는 방식	로킹 단위	오버헤드	로크 수	병행 수준	DB 공유도	커짐	감소	적어짐	낮아짐	감소	작아짐	증가	증가함	높아짐	증가
로킹 단위	오버헤드	로크 수	병행 수준	DB 공유도												
커짐	감소	적어짐	낮아짐	감소												
작아짐	증가	증가함	높아짐	증가												

• 트랜잭션 상태

트랜잭션 상태	설명
완료 상태 (Committed)	트랜잭션 작업 결과가 확정된 상태
활동 (Active)	트랜잭션이 시작되어 실행 중인 상태
실패 (Failed)	트랜잭션이 오류 등으로 비정상적으로 종료된 상태
철회 (Aborted)	트랜잭션이 Rollback 명령어에 의해 취소되는 경우
부분 완료 (Partially Committed)	트랜잭션이 Commit 명령어를 실행하고, 일부만 완료된 상태

완료 (Committed)	트랜잭션이 Commit 명령어를 실행하여 모든 작업이 성공적으로 완료되고, 영구적으로 저장된 상태
-------------------	--

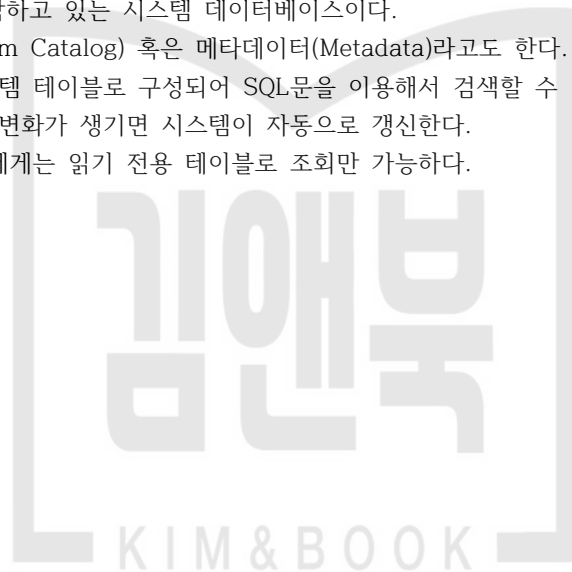
• 트랜잭션 제어

명령어	설명
커밋 (COMMIT)	- 트랜잭션의 작업을 영구적으로 데이터베이스에 반영 - 데이터가 영구적으로 유지됨
롤백 (ROLLBACK)	트랜잭션의 작업을 취소하고, 이전 상태로 되돌림
체크 포인트 (CHECKPOINT)	트랜잭션 내에 ROLLBACK 할 시점을 설정

059 데이터 사전★★

• 데이터 사전의 특징

- DBMS에서 필요로 하는 여러 가지 객체(기본 테이블, 뷰, 인덱스, 데이터베이스, 패키지, 접근 권한 등)에 관한 정보를 포함하고 있는 시스템 데이터베이스이다.
- 시스템 카탈로그(System Catalog) 혹은 메타데이터(Metadata)라고도 한다.
- 데이터 사전 또한 시스템 테이블로 구성되어 SQL문을 이용해서 검색할 수 있다.
- SQL문 등으로 개체에 변화가 생기면 시스템이 자동으로 갱신한다.
- 데이터 사전은 사용자에게는 읽기 전용 테이블로 조회만 가능하다.



CHAPTER 03 논리 데이터베이스 설계

060 데이터베이스 설계★★★

• 데이터베이스 설계

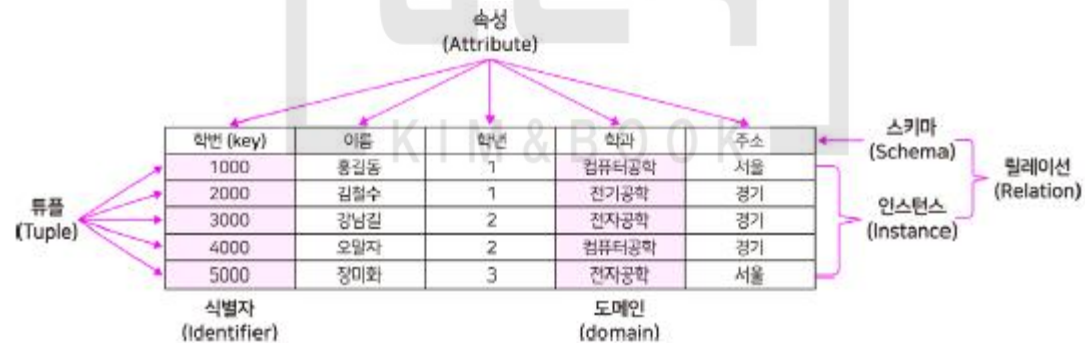
순서	단계	설명
1	요구조건 분석	<ul style="list-style-type: none"> - 데이터베이스의 사용 목적 파악 - 데이터베이스 구조 설계에 필요한 개체, 속성, 관계 제약조건 등을 식별
2	개념적 설계	<ul style="list-style-type: none"> - 정보를 구조화하기 위해 추상적 개념으로 독립적인 개념스키마를 설계 - 트랜잭션 모델링과 개념스키마 모델링 - 요구조건 분석을 통해 E-R 다이어그램을 작성
3	논리적 설계	<ul style="list-style-type: none"> - 컴퓨터가 이해할 수 있도록 DBMS에 맞게 논리적 자료구조로 사람이 이해하기 쉽게 변환 - 스키마를 평가 및 정제 - 정규화를 수행 - 트랜잭션의 인터페이스를 설계 - 테이블을 설계하는 단계에서 정규화
4	물리적 설계	<ul style="list-style-type: none"> - 논리적 구조로 표현된 데이터를 물리적 구조의 데이터로 DB에 변환 - 저장 레코드의 양식 설계함(데이터 타입, 데이터값의 분포, 접근 빈도) - 저장구조 및 접근 경로를 설정하고 레코드 집중의 분석 및 설계

• 데이터베이스 모델

구성 요소	설명
구조(Structure)	논리적인 개체 타입 간의 관계, 데이터 구조 및 정적 성질을 표현
연산(Operation)	조작하는 기본 도구로 실제 데이터를 처리하는 작업에 대한 명세
제약조건(Constraint)	DB에 저장될 수 있는 실제 데이터의 논리적인 제약 조건

061 관계 데이터 모델 구성요소★★★

• 관계 데이터 모델 구성요소



구성요소	설명
릴레이션(Relation)	<p>행(Row)과 열(Column)로 구성된 테이블</p> <p>릴레이션의 특징</p> <ul style="list-style-type: none"> - 속성은 단일 값을 가짐 - 튜플의 삽입, 삭제로 인해 릴레이션의 시간에 따라 변함 - 속성은 서로 다른 이름을 가짐 - 한 속성의 값은 모두 같은 도메인 값을 가짐 - 튜플의 순서는 상관없음 - 속성값은 모두 원자값으로 저장
속성(Attribute)	릴레이션의 세로 값으로 열(Column)이라고도 함
튜플(Tuple)	릴레이션의 가로 값으로 행(Row)이라고도 함

차수(Degree)	속성의 수
카디널리티(Cardinality)	튜플의 수
인스턴스(Instance)	- 어느 한 시점 릴레이션에 존재하는 튜플들의 집합 - 외연(relation extension)으로도 불림
스키마(Schema)	릴레이션의 구성 및 정보에 대한 기본적인 구조를 정의함
도메인(Domain)	하나의 속성(Attribute)가 취할 수 있는 원자값들의 집합
식별자(Identifier)	속성(Attribute) 중에서 튜플을 유일하게 식별할 수 있는 속성(Attribute)

062 개체-관계(E-R) 모델★★★

- E-R 다이어그램

기호(이름)	설명
 (사각형)	- 개체(Entity) - 여러 개의 속성을 가지며, 동일한 개체는 존재할 수 없음
 (마름모)	- 관계(Relationship) - 개체들이 가지는 관계로 1:1, 1:N, N:M 등 다양한 관계를 표현 가능함
 (속성)	- 속성(Attribute) - 개체의 속성을 표현함
 (이중 타원)	- 다중 값 속성 - 하나의 독립적인 속성이나 그 안에 여러 개의 값이 포함될 수 있음
 (선)	선, 링크

063 키(Key) 종류★★★

- 키(Key) 종류

종류	설명
기본키 (Primary Key)	- 후보키 중에서 선정된 주키(Main Key)로 중복된 값과 NULL 값을 가질 수 없음 - 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성
후보키 (Candidate Key)	- 릴레이션에서 튜플을 유일하게 구별해주는 속성 또는 속성들의 조합 - 기본키로 사용할 수 있는 속성들로 유일성과 최소성을 만족함 - 모든 릴레이션에는 반드시 하나 이상의 후보키가 존재 해야 함 - 기본키와 대체키를 합친 키(기본키+후보키, 대체키+후보키)
대체키 (Alternate Key)	후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키로 보조키로도 불림
슈퍼키 (Super Key)	- 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키 - 릴레이션에 있는 모든 튜플에 대해 유일성은 만족시키지만 최소성은 만족시키지 못하는 키
외래키 (Foreign Key)	- 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합으로 릴레이션 간의 참조 관계를 표현함

064 데이터베이스 무결성★★★

• 무결성의 종류

종류	설명
개체 무결성 (Entity Integrity)	- 기본 키 - 중복 값을 가질 수 없음 - NULL 값이 될 수 없음
참조 무결성 (Referential Integrity)	- 외래 키 - 외래키 값은 NULL 값이거나 참조 릴레이션의 기본 키와 동일해야 함 - 참조되는 튜플이 반드시 존재해야 함
사용자 정의 무결성 (User-Defined. Integrity)	사용자가 정의한 조건에 만족해야 함

065 관계 데이터 언어★★★

• 일반 집합 연산자

연산자	기호/표현	설명
합집합 (UNION)	- 기호: \cup - $R \cup S$	- 두 릴레이션 R과 S의 합집합 - 두 릴레이션의 합이 추출되고, 중복은 하나만 포함됨
교집합 (INTERSECTION)	- 기호: \cap - $R \cap S$	- 두 릴레이션 R과 S의 교집합 - 두 릴레이션의 중복되는 값만 추출됨
차집합 (DIFFERENCE)	- 기호: $-$ - $R - S$	- 두 릴레이션에서 R 릴레이션에만 존재하는 값 - R 릴레이션에서 S 릴레이션에 중복되지 않는 값만 추출됨
교차곱 (CARTESIAN PRODUCT)	- 기호: \times - $R \times S$	- 두 릴레이션의 가능한 모든 튜플들의 집합 - 두 릴레이션의 가능한 모든 조합을 추출함

• 순수 관계 연산자

연산자	기호/표현	설명
선택 (Select)	- 기호: σ - $\sigma_{\langle \text{조건} \rangle}(R)$	- 릴레이션 R에서 조건을 만족하는 튜플 반환 - 수평 연산
추출 (Project)	- 기호: π - $\pi_{\langle \text{속성리스트} \rangle}(R)$	- 릴레이션 R에서 중복을 제거한 속성들의 값을 반환 - 수직 연산
조인 (Join)	- 기호: \bowtie - $R \bowtie S$	두 릴레이션이 공통으로 가지고 있는 속성을 이용해 하나의 릴레이션을 만들어 튜플을 반환
나누기 (Division)	- 기호: \div - $R \div S$	S 릴레이션의 속성 도메인 값과 일치하는 R 릴레이션의 S를 속성을 제외한 튜플을 반환

• 논리 기호

구분	설명								
연산자	<table><tr><td>연산자</td><td>연산</td></tr><tr><td>\vee</td><td>OR</td></tr><tr><td>\wedge</td><td>AND</td></tr><tr><td>\neg</td><td>NOT</td></tr></table>	연산자	연산	\vee	OR	\wedge	AND	\neg	NOT
연산자	연산								
\vee	OR								
\wedge	AND								
\neg	NOT								
정량자	<table><tr><td>연산자</td><td>연산</td></tr><tr><td>\forall</td><td>- 가능한 모든 튜플 - 모든 것에 대하여(for all)</td></tr><tr><td>\exists</td><td>- 하나라도 일치하는 튜플 - 존재한다(There exist)</td></tr></table>	연산자	연산	\forall	- 가능한 모든 튜플 - 모든 것에 대하여(for all)	\exists	- 하나라도 일치하는 튜플 - 존재한다(There exist)		
연산자	연산								
\forall	- 가능한 모든 튜플 - 모든 것에 대하여(for all)								
\exists	- 하나라도 일치하는 튜플 - 존재한다(There exist)								

066 데이터베이스 정규화(Normalization)와 반정규화(Denormalization)★★★

• 정규화

개념	설명								
정규화 목적	<ul style="list-style-type: none"> - 중복 데이터를 최소화해 삽입, 삭제, 갱신 이상들을 제거하기 위함 - 수정, 삭제 시 이상 현상의 최소화 - 테이블 불일치 위험의 최소화 - 중복을 배제하여 삽입, 삭제, 갱신 이상의 발생을 방지 - 데이터 구조의 안정성을 최대화 - 데이터 삽입 시 릴레이션의 재구성 필요성을 최소화 - 데이터베이스 검색 시 효율성을 최대화 								
이상	<ul style="list-style-type: none"> - 정규화를 거치지 않으면 생기는 곤란한 현상 - 속성들에 존재하는 여러 종류의 종속관계를 하나의 릴레이션에 표현할 때 발생 <table border="1"> <thead> <tr> <th>이상 현상</th><th>설명</th></tr> </thead> <tbody> <tr> <td>삽입 이상 (Insertion Anomaly)</td><td>릴레이션에서 데이터를 삽입할 때 의도와는 상관없이 원하지 않는 값들도 함께 삽입되는 현상</td></tr> <tr> <td>갱신 이상 (Update Anomaly)</td><td>릴레이션에서 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상</td></tr> <tr> <td>삭제 이상 (Deletion Anomaly)</td><td>릴레이션에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는 연쇄 삭제 현상</td></tr> </tbody> </table>	이상 현상	설명	삽입 이상 (Insertion Anomaly)	릴레이션에서 데이터를 삽입할 때 의도와는 상관없이 원하지 않는 값들도 함께 삽입되는 현상	갱신 이상 (Update Anomaly)	릴레이션에서 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상	삭제 이상 (Deletion Anomaly)	릴레이션에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는 연쇄 삭제 현상
이상 현상	설명								
삽입 이상 (Insertion Anomaly)	릴레이션에서 데이터를 삽입할 때 의도와는 상관없이 원하지 않는 값들도 함께 삽입되는 현상								
갱신 이상 (Update Anomaly)	릴레이션에서 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상								
삭제 이상 (Deletion Anomaly)	릴레이션에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는 연쇄 삭제 현상								

• 정규화 단계

정규화 단계	조건
1정규형 (1NF)	<ul style="list-style-type: none"> - 도메인이 원자값으로 구성되어야 함 - 원자값이 아닌 도메인을 분해해야 함
2정규형 (2NF)	<ul style="list-style-type: none"> - 1정규형을 만족 - 부분 함수 종속성을 제거함 - 부분집합 중 원래 자신의 집합을 제외한 것으로 X와 Y를 각각 R의 애트리뷰트 집합의 부분 집합이라고 할 경우 $X \rightarrow Y$로 표시
3정규형 (3NF)	<ul style="list-style-type: none"> - 2정규형을 만족 - 이행 함수 종속을 제거 - $A \rightarrow B$이고, $B \rightarrow C$일 때 $A \rightarrow C$를 만족하는 관계
보이스-코드 정규형 (BCNF)	<ul style="list-style-type: none"> - 3정규형을 만족 - 기본키를 제외하고 후보키가 있는 경우 후보키가 기본키를 종속시키면 분해 (모든 결정자가 후보키 집합에 속해야 함)
4정규형 (4NF)	<ul style="list-style-type: none"> - BCNF를 만족 - 다치 종속을 제거 - 여러 컬럼들이 하나의 컬럼을 종속시키는 경우 분해하여 다중값(다치)을 제거
5정규형 (5NF)	<ul style="list-style-type: none"> - 4정규형을 만족 - 조인 종속성을 제거 - 조인에 의해서 종속성이 발생하는 경우 분해 - 기본키를 통하지 않는 조인 종속 제거

CHAPTER 04 물리 데이터베이스 설계

067 스토리지와 분산 데이터베이스★★★

• 스토리지

저장 장치	설명
DAS (Direct Attached storage)	<ul style="list-style-type: none"> - 직접 연결 장치 - 직접 연결로 속도가 빠르고 안전함 - 직접 연결로 저장장치 확장 및 유연성이 떨어짐
NAS (Network Attached Storage)	<ul style="list-style-type: none"> - 네트워크 결합 장치 - 네트워크 이용이 많은 경우 성능저하 발생 - 파일 전송의 유연성이 높음
SAN (Storage Area Network)	<ul style="list-style-type: none"> - 광 케이블(Fiber Channel)을 통한 결합 장치 - 높은 성능을 보장
SDS (Software-defined storage)	<ul style="list-style-type: none"> - 소프트웨어로 전체 스토리지 자원을 관리 - 하나의 저장장치처럼 사용할 수 있도록 함

• 분산 데이터베이스의 장점 및 단점

장점	단점
<ul style="list-style-type: none"> - 분산 제어가 용이 - 효율성과 융통성이 높음 - 시스템 확장이 용이 - 지역 자치성이 높음 	<ul style="list-style-type: none"> - 설계가 비교적 어려움 - 개발 비용과 처리 비용이 증가

• 분산 데이터베이스 시스템 구성

구성	설명
분산 처리기	물리적으로 분산되어 지역별로 필요한 데이터를 할 수 있는 지역 컴퓨터를 분산 처리기라고 함
분산 데이터베이스	지리적으로 분산된 데이터베이스로서 해당 지역의 특성에 맞게 데이터베이스를 구성함
통신 네트워크	분산 처리기들을 통신망으로 연결하여 논리적으로 하나의 시스템처럼 작동할 수 있도록 함

• 분산 데이터베이스 목표

투명성	설명
위치 투명성 (Location Transparency)	사용자가 데이터베이스의 하드웨어와 소프트웨어의 물리적 위치를 알 필요가 없음
중복 투명성 (Replication Transparency)	동일 데이터가 중복되어 여러 곳에 있어도 사용자에게 하나만 존재하도록 인식함
병행 투명성 (Concurrency Transparency)	다수의 트랜잭션이 동시에 실행되더라도 그 결과는 영향을 받지 않음
장애 투명성 (Failure Transparency)	분산 데이터베이스 시스템의 이상이 발생해도 트랜잭션을 정확하게 처리해야 함

068 데이터베이스 이중화와 암호화★★

• 데이터베이스 암호화

용어	설명
암호화 (Encryption)	암호화되지 않은 평문을 암호문으로 바꿈
복호화 (Decryption)	암호화된 암호문을 암호화되지 않은 평문으로 바꿈
키 (Key)	적절한 암호화 및 복호화를 위해 사용하는 값

069 파티셔닝★★

• 파티셔닝(Partitioning)

유형	설명
범위 분할 (Range Partitioning)	지정한 열의 값을 기준으로 분할함
해시 분할 (Hash Partitioning)	해시 함수를 적용한 결과 값에 따라 데이터를 분할함
목록 분할 (List Partitioning)	값 목록에 파티션을 할당하여 분할함
조합 분할 (Composite Partitioning)	범위 분할, 해시 분할, 목록 분할 중 2개 이상의 파티셔닝을 결합함
라운드 로빈 (Round Robin Partitioning)	레코드를 균일하게 분배하는 방식임

