

# SQLD

## 2024년 자격증 특강

컴퓨터·AI공학과  
김윤수 교수  
(kchris000@gmail.com)



# SQLD

데이터자격검정 : <https://www.dataq.or.kr/>

## SQLD

SQLD (SQL Developer, SQL개발자)는 데이터베이스와 데이터 모델링에 대한 지식을 바탕으로 응용 소프트웨어를 개발하면서 데이터를 조작하고 추출하는데 있어서 정확하고 최적의 성능을 발휘하는 SQL을 작성할 수 있는 개발자를 의미합니다.

## SQLD 자격 시험 - 필기 (선택형 50문항, 문항당 2점)

| 과목명         | 과목별 세부 항목                  | 문항수 | 배점  | 검정시간 | 합격기준                      |
|-------------|----------------------------|-----|-----|------|---------------------------|
| 데이터 모델링의 이해 | 데이터 모델링의 이해<br>데이터 모델과 SQL | 10  | 20점 | 90분  | 60점 이상<br>(과목별 40% 미만 과락) |
| SQL 기본 및 활용 | SQL 기본<br>SQL 활용<br>관리 구문  | 40  | 80점 |      |                           |

# SQLD

## SQLD 출제기준

### ✓ 데이터 모델링의 이해

- 데이터 모델링의 이해 : 데이터모델의 이해, 엔터티, 속성, 관계, 식별자
- 데이터 모델과 SQL : 정규화, 관계와 조인의 이해, 모델이 표현하는 트랜잭션의 이해, Null 속성의 이해, 본질식별자 vs 인조식별자

### ✓ SQL 기본 및 활용

- SQL 기본 : 관계형 데이터베이스 개요, SELECT 문, 함수, WHERE 절, GROUP BY, HAVING 절, ORDER BY 절, 조인, 표준 조인
- SQL 활용 : 서브 쿼리, 집합 연산자, 그룹 함수, 윈도우 함수, Top N 쿼리, 계층형 질의와 셀프 조인, PIVOT 절과 UNPIVOT 절, 정규 표현식
- 관리 구문 : DML, TCL, DDL, DCL

# SQLD

## 자격증 특강

컴퓨터·AI공학과  
김윤수 교수



# 01. 데이터 모델링의 이해

## 데이터 모델링의 이해

- ✓ 데이터 모델의 이해
- ✓ 엔터티
- ✓ 속성
- ✓ 관계
- ✓ 식별자

# 01. 데이터 모델링의 이해

## 데이터 모델의 이해

### 모델링 (Modeling)의 개요

#### ✓ 모델링의 개념

- 현실 세계 속의 다양한 데이터 들의 공통적인 특징을 찾아서 상위 레벨의 표현 기법으로 정리하는 과정 및 기법
- 현실 세계를 단순화하여 표현하는 과정 혹은 기법
  - ▶ 현실 세계의 데이터를 추상화 하여 물리적 데이터베이스로 변경하는 과정

#### ✓ 모델링의 일반적인 특징

- 추상화 (Abstraction), 단순화(Simplification), 명확성 (Clarity)
- 데이터 관점, 프로세스 관점, 데이터와 프로세스 관점
  - ▶ 잘 정의된 표기법에 의해 표현되어야 한다.
  - ▶ 주의사항 : 중복, 비유연성, 비일관성

# 01. 데이터 모델링의 이해

## 데이터 모델의 이해

### 모델링(Modeling)의 구분 - 3단계 모델링

#### ✓ 3단계 모델링

① 개념적 데이터 모델링    ② 논리적 데이터 모델링    ③ 물리적 데이터 모델링

| 종류 (단계) | 설명   |
|---------|--|
| 개념적 모델링 | <ul style="list-style-type: none"><li>- 요구사항에 기반한 정보 구조의 모형을 변화하여 일반화</li><li>- 핵심 엔티티와 엔티티간 상호관계를 표현하기 위한 ERD(ER Diagram)을 작성</li><li>- 정보시스템에 어떤 방식으로 적용할지 검토하며, 의사소통을 위한 가시성 확보</li></ul> |
| 논리적 모델링 | <ul style="list-style-type: none"><li>- 개념적 모델링에서 도출한 관계를 구조적으로 설계</li><li>- 데이터 이상현상 제거를 위한 정규화 수행</li><li>- 식별자 정의, M:M 관계 제거, 참조 무결성 정의 등을 수행</li></ul>                                   |
| 물리적 모델링 | <ul style="list-style-type: none"><li>- 논리적 데이터 모델을 DBMS의 특징에 맞도록 데이터베이스 스키마를 구축</li><li>- 데이터의 물리적 위치, 인덱스, 파티션, 분산 구조 등을 결정</li><li>- 성능을 고려한 반정규화 검토</li></ul>                            |

# 01. 데이터 모델링의 이해

## 데이터 모델의 이해

### 모델링 (Modeling)의 구분 - 3단계 모델링 (개념적)

#### ✓ 개념적 모델링

- 주제영역 도출, 핵심 엔티티 도출, 관계설정으로 구분

| 종류        | 설명  |
|-----------|---|
| 주제영역 도출   | <ul style="list-style-type: none"><li>- 요구사항에 따라 관리하고자 하는 데이터 그룹 정의</li><li>- 일반적으로 관련 업무 기능(Function)이 존재, 보편적으로 사용하는 업무 용어 사용</li><li>- 예) 상품, 판매, 인사</li></ul> |
| 핵심 엔티티 도출 | <ul style="list-style-type: none"><li>- 관리하고자 하는 정보의 최소 관리단위 도출</li><li>- 데이터베이스 테이블로 구현되는 중심 객체</li><li>- 예) 구매 내역, 사원 마스터</li></ul>                             |
| 관계 설정     | <ul style="list-style-type: none"><li>- 핵심 엔티티 간 관계 파악, 양방향 로직(규칙)을 표현</li><li>- 데이터베이스의 외부키(Foreign Key)로 구체화</li><li>- 예) 구매한다, 소속된다, 포함된다,</li></ul>           |



# 01. 데이터 모델링의 이해

## 데이터 모델의 이해

### 모델링(Modeling)의 구분 - 3단계 모델링 (논리적)

#### ✓ 논리적 모델링

- 식별자 선정, 관계 설정/해소, 다중값 속성 해소, 정규화 등으로 구분

| 단계         | 설명   |
|------------|--|
| 모든 엔티티 도출  | - 요구사항 분석, 업무와 관련된 모든 엔티티를 도출                              |
| 식별자 선정     | - 엔티티를 대표하는 식별자(Key) 선정                                    |
| 엔티티 정제     | - 식별자가 동일하거나 의미가 유사한 엔티티를 통합하거나 분리하여 구조화                   |
| 관계 설정/해소   | - 1:1 관계, 1:N 관계, N:1 관계를 설정.<br>- M:N 관계를 교차 엔티티를 이용하여 해소 |
| 다중값 속성 해소  | - 다중값 속성을 독립 속성 혹은 세부 항목으로 분류할지 검토 및 결정                    |
| 정규화 및 제약조건 | - 데이터 중복제거 및 이상현상 해소를 위한 정규화 수행<br>- 무결성 검사 및 제약 수행        |

# 01. 데이터 모델링의 이해

## 데이터 모델의 이해

### 모델링 (Modeling)의 구분 - 3단계 모델링 (물리적)

#### ✓ 물리적 모델링

- 실제 사용할 DBMS의 특징에 따라 논리적 모델의 속성 타입과 크기 등을 결정, 릴레이션과 컬럼의 제약조건 정의
- 물리적 데이터베이스 구축을 위한 준비 : 물리적 순서, 성능향상 전략, 공간할당 정책, 백업/복구 전략
- 물리적 데이터베이스 구축 : 테이블 구조 설계 및 생성, 인덱스 설계 및 생성, 성능향상을 위한 반정규화

#### ✓ 반정규화

- 성능과 효율성을 위하여 데이터 중복을 허용, 조인으로 인한 성능 저하가 예상될 때 사용
- '테이블 추가 및 분할, 중복 컬럼 활용' 등을 고려

# 01. 데이터 모델링의 이해

## 데이터 모델의 이해

### 모델링(Modeling)의 구분 - 3단계 스키마

#### ✓ 3단계 스키마

- ANSI-SPARC(American National Standards Institute, Standards Planning And Requirements Committee)가 제안한 설계 표준. 독립성 보장
- 외부 스키마 (External Schema) : 사용자 관점
- 개념 스키마 (Conceptual Schema) : 통합 데이터 관점
- 내부 스키마 (Internal Schema) : 물리적 관점

※ 사상(Mapping) : 상호 독립적인 개념을 연결시켜주는 역할 혹은 매개체

# 01. 데이터 모델링의 이해

## 데이터 모델의 이해

### ERD (Entity Relationship Diagram)

#### ✓ ERD의 개념

- 관심 대상이 되는 데이터를 형식화된 다이어그램을 사용하여 표현하는 방법
- 실세계의 데이터를 개체(Entity type)과 개체간 상호관계(Relationship)으로 표현
- 초기 ER모델 : 개체(Entity), 관계(Relationship), 속성(Attribute) 개념으로 구성
- 확장 ER모델 : 일반화, 계층, 복합속성 등의 개념이 추가됨

#### ※ ERD를 이용한 모델링 방법:

엔터티 그리기 → 엔터티 배치 → 엔터티 간의 관계설정 → 관계명 기술 → 관계의 참여도 기술 → 관계의 필수여부기술

# 01. 데이터 모델링의 이해

## 데이터 모델의 이해




### ERD (Entity Relationship Diagram) - 표기법

#### ✓ ERD의 구성요소

<참고 및 추가 학습>

- ERD의 표기법은 다양합니다.

- **Peter Chen, IDEFIX** (ERWin), **IE/Crows' Foot**, UML 등

| 구성요소                 | 설명   | 표기법   |
|----------------------|--|---|
| 개체<br>(Entity)       | <ul style="list-style-type: none"><li>- 현실 세계 혹은 개념적으로 존재하는 대상 중 핵심이 되는 정보</li><li>- Entity Instance는 Entity가 구체적인 정보로 실현된 것</li></ul> |    |
| 관계<br>(Relationship) | <ul style="list-style-type: none"><li>- 개체간 존재하는 의미 있는 연관성</li><li>- 개체간 존재 형태 혹은 행위로 상호 영향을 주고 받음</li></ul>                           |   |
| 속성<br>(Attribute)    | <ul style="list-style-type: none"><li>- 개체의 특성이나 상태의 세부 항목</li><li>- 인스턴스를 다른 인스턴스와 구별할 수 있도록 하는 속성을 식별자 (Identifier)라고 함</li></ul>    |  |

# 01. 데이터 모델링의 이해

## 엔터티

### 엔터티 (Entity)의 개념과 특징

#### ✓ 엔터티의 개념과 특징

- 현실 세계 혹은 개념적으로 존재하는 대상 중 핵심이 되는 정보
- 식별이 가능한 실체, 객체 (명확성)
  - ▶ 주제와 연관된 정보, 여러 개의 인스턴스 유지, 서로 구분할 수 있는 정보 (식별자), 속성 및 관계를 가짐

#### ✓ 엔터티의 분류

- 형태에 따른 분류 : 유형 엔터티, 개념 엔터티, 사건 엔터티
- 시점에 따른 분류 : 기본 엔터티, 중심 엔터티, 행위 엔터티
  - ▶ 엔터티의 명칭은 업무 목적에 따른 자연스러운 이름을 부여한다.

# 01. 데이터 모델링의 이해

## 속성

### 속성(Attribute)의 개념과 특징

#### ✓ 속성의 개념과 특징

- 개체의 특성이나 상태의 세부 항목
- 엔터티의 특성과 상태를 의미하는 최소한의 구성 단위

#### ✓ 속성의 다양한 형식

- 단순 속성 (Simple Attribute) : 더 이상 분해할 수 없는 단위의 속성
- 복합 속성 (Composite Attribute) : 2개 이상의 기본 속성들로 분해 가능한 속성
- 단일치 속성 (Single Valued Attribute) : 하나의 값만 존재하는 속성
- 다중치 속성 (Multi Valued Attribute) : 여러 개의 값을 가질 수 있는 속성

▶ 기본속성, 설계속성, 파생속성 으로 구분할 수도 있음

#### <참고>

- PK (Primary Key) 속성, FK (Foreign Key) 속성, 일반 속성
- 도메인 : 속성이 가질 수 있는 속성값의 범위

#### <시험 확인>

- 한 개의 속성은 한 개의 속성값을 갖는다.

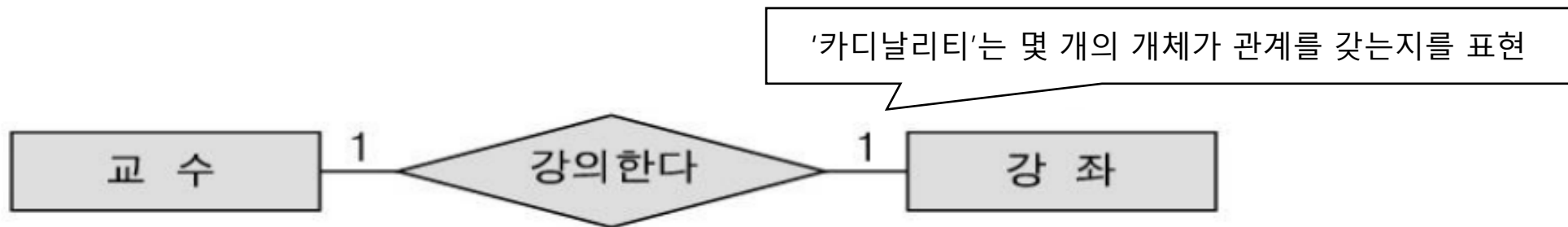
# 01. 데이터 모델링의 이해

## 관계

### 관계(Relation)의 개념과 특징

#### ✓ 관계의 개념과 특징

- 엔터티(개체)간 존재하는 의미 있는 연관성
- 존재 관계와 행위 관계로 구분할 수 있음
- 표현 방법 : 관계명, 관계차수(Cardinality), 선택여부(필수, 선택)
- 관계 형태 : 일반관계, 자기 참조 관계, 병렬 관계, 수퍼타입/서브타입 관계





# 01. 데이터 모델링의 이해

## 식별자

### 식별자(Identifier)의 개념과 특징

#### ✓ 식별자의 개념과 특징

- 하나의 Relation에서 Tuple을 유일하게 식별할 수 있는 속성 (혹은 속성의 집합)
- 유일성, 최소성, 대표성, 불변성, 존재성
  - 유일성 : Tuple 들을 구분할 수 있는 특징 (Unique, Not Null)
  - 최소성 : 유일성을 갖는 최소한의 속성으로 구성
  - 대표성 : 해당 Relation을 대표
  - 불변성 : (가능한) 변하지 않은 값
  - 존재성 : NULL이 될 수 없음

# 01. 데이터 모델링의 이해

## 식별자

### 식별자(Identifier)의 종류

#### ✓ 식별자의 종류

- 주식별자 (Primary) vs. 보조식별자 (Alternative)
- 내부 식별자 (Internal) vs. 외부 식별자 (Foreign)
- 단일 식별자 (Single) vs. 복합 식별자 (Composite)
- 원조 식별자 (Original) vs. 대리 식별자 (Surrogate)

<확인>

- 식별 관계 : (부모 엔터티)식별자가 다른(자식) 개체의 주식별자로 사용
- 비식별 관계 : (부모 엔터티 )식별자가 다른(자식) 개체의 일반 속성으로 사용

# 01. 데이터 모델링의 이해

## 식별자

### 식별자(Identifier)의 종류

<참고>

- 논리적인 관점에서 식별자라고 하며,  
물리적 관점의 용어로 키(Key)를 이용함

#### ✓ 식별자의 종류


| 종류                       | 설명   |
|--------------------------|--|
| 기본키<br>(Primary Key)     | - 주식별자, 여러 개의 후보키 중에서 하나를 선정, 테이블을 대표하는 키<br>- 대표성 |
| 후보키<br>(Candidate key)   | - 키의 특성인 유일성과 최소성을 만족하는 키                          |
| 슈퍼키<br>(Super key)       | - 유일성은 만족하나 최소성을 만족하지 않는 키                         |
| 대체키<br>(Alternative Key) | - 후보키 중 기본키로 선정되지 않는 키, 기본키를 대체할 수 있는 키            |
| 외래키<br>(Foreign Key)     | - 한 릴레이션의 속성(집합)이 다른 릴레이션의 기본키로 이용되는 키             |

# 01. 데이터 모델링의 이해

## 추가 정리

### ERD와 엔터티, 속성, 관계, 식별자

- ✓ 엔터티(Entity, 개체) : 표현할 정보를 갖고 있는 독립적인 실체

 (Regular, Strong) entity type

 Weak entity type

- ✓ 속성(Attribute) : 개체가 갖고 있는 특징

 Attribute

 Multivalued attribute

 Key attribute

 Partial Key Attribute

 Composite attribute

 Derived attribute

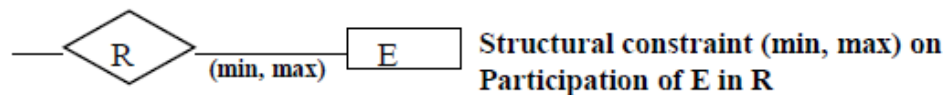
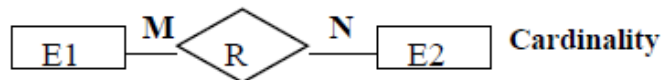
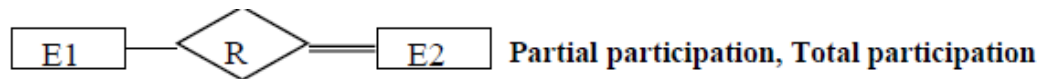
# 01. 데이터 모델링의 이해

## 추가 정리

### ERD와 엔터티, 속성, 관계, 식별자

#### ✓ 관계 : 개체들 간의 관계

- 강한 관계 : 하나의 개체가 다른 개체를 통해 존재할 수 있는 의존적 관계, 마름모로 표현
- 약한 관계 : 다른 객체와 독립적으로 존재할 수 있는 독립적 관계, 이중 마름모로 표현



# 01. 데이터 모델링의 이해

## 추가 정리

### 관계의 변환 (논리적 모델링)

#### ✓ 관계 변환 방법

- 논리적 모델링 단계 : 식별자 선정, 관계 설정/해소, 다중값 속성 해소, 정규화 등을 수행
- 엔티티와 속성 : 테이블과 컬럼으로 변환
- 키 : 식별자는 기본키로, 관계를 외래키로 변환
- 1:N 관계 : 1측에 해당하는 기본키를 N측 테이블의 외래키로 구성
- M:N 해소 : 구성하는 엔티티들의 기본키로 구성된 별도의 테이블 생성
- 다중값 속성 : 해당 엔티티의 기본키와 다중값 속성을 포함한 키를 가진 새로운 엔티티 생성
- 복합 속성 : 하나의 엔티티로 유지하거나 분할하여 관리

# SQLD

## 자격증 특강

컴퓨터·AI공학과  
김윤수 교수



## 02. 관계형 데이터베이스와 관리구문

### 관계형 데이터베이스와 관리구문

- ✓ 관계형 데이터베이스 개요
- ✓ 관리 구문
  - DML
  - TCL
  - DDL
  - DCL



## 02. 관계형 데이터베이스와 관리구문

### 관계형 데이터베이스 개요

#### 데이터와 데이터베이스(Database)의 개념

##### ✓ 데이터(Data)의 개념

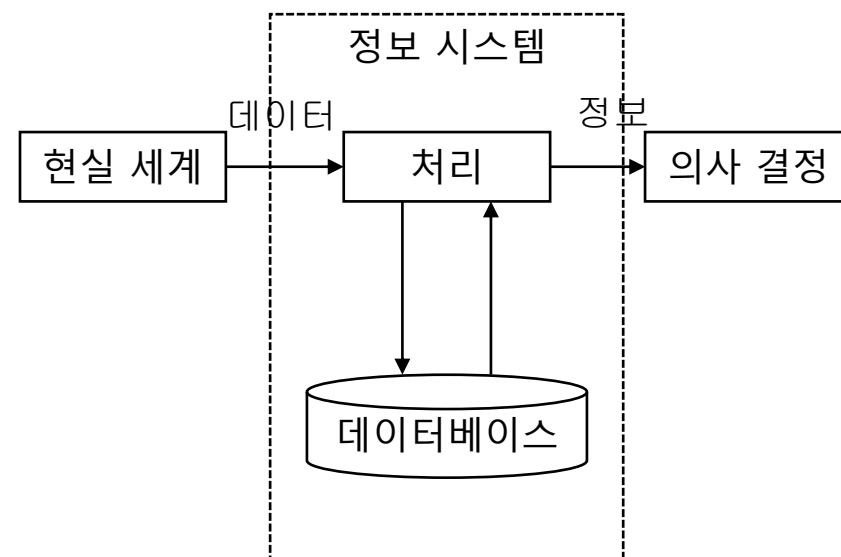
- 관찰 혹은 측정하여 수집한 사실(fact)이나 값(value)
- 수치 데이터, 문자 데이터, 이미지 데이터 등

##### ✓ 정보(Information)의 개념

- 데이터를 의사 결정 등에 활용하기 위하여 체계적으로 처리한 결과물
- 일정한 프로그램에 의해 처리되거나 정의된 양식에 의하여 가공됨

##### ✓ 데이터베이스(Database)의 개념

- 조직, 업무 목적에 따라 여러 사람들이 공유하여 사용할 목적으로 통합 및 관리되는 데이터의 집합
- 공동 목적을 지원하기 위한 관련된 자료들의 집합체



## 02. 관계형 데이터베이스와 관리구문

### 관계형 데이터베이스 개요

#### 데이터베이스의 특징

##### ✓ 데이터베이스의 특징 (1)

| 구성 요소                       | 설명                                 |
|-----------------------------|------------------------------------|
| 통합 데이터<br>(Integrated Data) | - 중복을 제외하거나 의도적인 중복을 파악하여 관리       |
| 저장 데이터<br>(Stored Data)     | - 컴퓨터 시스템의 저장매체에 저장하여 관리           |
| 운영 데이터<br>(Operation Data)  | - 조직의 기능과 역할 수행을 위해 필요한 데이터        |
| 공용 데이터<br>(Shared Data)     | - 여러 사용자 혹은 응용시스템들이 목적에 따라 공유하여 이용 |

## 02. 관계형 데이터베이스와 관리구문

### 관계형 데이터베이스 개요

#### 데이터베이스의 특징

##### ✓ 데이터베이스의 특징 (2)

| 특징                                   | 설명   |
|--------------------------------------|--|
| 실시간 접근성<br>(real time accessibility) | - 사용자의 질의에 대하여 즉시 처리 및 응답  |
| 계속적인 진화<br>(continuous evolution)    | - 삽입, 삭제, 갱신을 통하여 항상 최근의 정확한 데이터를 동적으로 유지                              |
| 동시 공유<br>(concurrent sharing)        | - 여러 사용자가 동시에 원하는 데이터를 공유  |
| 내용에 의한 참조<br>(content reference)     | - 데이터베이스에 있는 데이터를 참조할 때 튜플(tuple)의 주소나 위치가 아닌 사용자가 요구 하는 데이터 내용에 따라 참조 |
| 데이터 논리적 독립성<br>(independence)        | - 응용프로그램과 데이터베이스를 독립시킴으로써 데이터 논리적 구조를 변경시키더라도 응용프로그램은 변경되지 않음          |

## 02. 관계형 데이터베이스와 관리구문

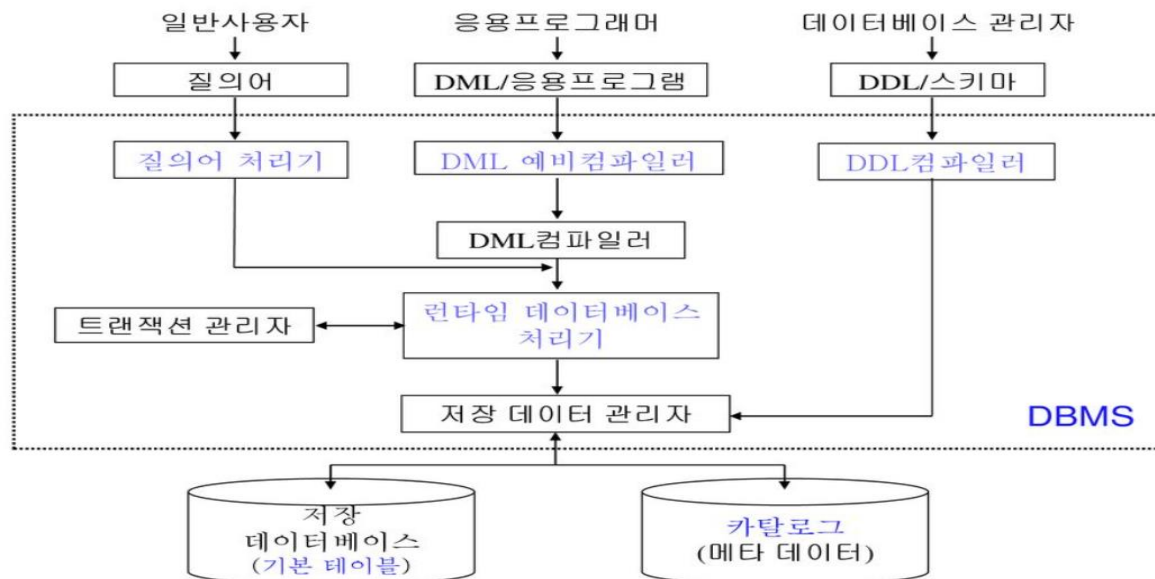
### 관계형 데이터베이스 개요

#### 데이터베이스 관리시스템의 개념과 특징

##### ✓ DBMS (DataBase Management System)의 개념

- 사용자와 데이터베이스 사이에서 데이터를 관리하고 사용자의 요청에 따라 데이터베이스에 연산을 수행하여 정보를 생성하는 프로그램

##### ✓ 관계형 DBMS의 구성도



## 02. 관계형 데이터베이스와 관리구문

### 관계형 데이터베이스 개요

#### DBMS의 다양한 유형

##### ✓ DBMS의 다양한 유형과 특징

| DBMS 유형                | 설명  |
|------------------------|---|
| 계층형 데이터베이스<br>(DBMS)   | <ul style="list-style-type: none"><li>- 데이터의 관계를 트리 구조로 정의, 부모-자식 형태를 갖는 구조이다.</li><li>- 상위 레코드가 복수의 하위 레코드를 갖는다.</li><li>- 데이터 중복의 단점이 있다.</li></ul> |
| 네트워크형 데이터베이스<br>(DBMS) | <ul style="list-style-type: none"><li>- 계층형 데이터의 데이터 중복 문제를 해결한 구조로, 레코드 간의 다양한 관계를 그물처럼 갖는다.</li><li>- 복잡한 구조로 인하여 구조 변경 시 어려움이 따른다.</li></ul>       |
| 관계형 데이터베이스<br>(DBMS)   | <ul style="list-style-type: none"><li>- 행/열로 구성된 Table 간의 관계를 표현한다.</li><li>- SQL을 이용하여 데이터에 접근, 관리한다.</li><li>- 오라클, MySQL/MariaDB 등</li></ul>       |
| NoSQL                  | <ul style="list-style-type: none"><li>- Document, Key-Value의 비정형 구조를 갖는다.</li><li>- 전용 API를 이용하여 데이터에 접근, 관리한다.</li><li>- 카산드라, 몽고 DB 등</li></ul>     |

## 02. 관계형 데이터베이스와 관리구문

### 관계형 데이터베이스 개요

#### 관계형 DBMS의 구성요소

##### ✓ 관계형 DBMS의 구성요소

| 구성 요소          | 설명   |
|----------------|--|
| DDL 컴파일러       | - DDL(Data Definition Language)로 명세된 스키마(schema)를 변환하여 카테고리에 저장                          |
| DML 컴파일러       | - DML(Data Manipulation Language)을 Object Code로 변환                                       |
| 질의어 처리기        | - Query Processor<br>- 질의문을 Parsing, Analysis, Compile하여 데이터베이스에 접근하기 위한 Object Code를 생성 |
| 런타임 데이터베이스 처리기 | - 데이터베이스 연산을 저장 데이터 관리자(Stored Data Manager)를 통해 수행, 실행 시간에 데이터베이스에 접근                   |

## 02. 관계형 데이터베이스와 관리구문

### SQL의 개요

#### SQL과 관계 대수

- ✓ SQL (Structured Query Language)의 개념
  - 관계대수에 기초하여 RDBMS의 데이터를 관리하기 위한 프로그램 언어
  - RDBMS에 저장된 데이터를 생성, 조작, 제어하기 위한 프로그램 언어
- ✓ 관계 대수의 개념
  - 원하는 결과를 위해 릴레이션을 처리하는 연산의 집합
  - 관계대수식(Relational Algebra Expression)은 연산을 수행하기 위한 식
  - 관계대수식은 단항 연산자와 이항 연산자로 구분

## 02. 관계형 데이터베이스와 관리구문

### SQL의 개요

#### 관계 연산자의 종류와 특징

##### ✓ 관계 연산자의 개념

- 원하는 결과를 위해 릴레이션을 처리하는 연산, 연산의 집합관계 대수의 개념

##### ✓ 관계 연산자 - 일반 집합 연산자의 종류와 특징

| 연산자                                     | 설명  |
|---|---|
| 합집합 (UNION, $\cup$ )                    | - 이항연산, 2개의 릴레이션을 합집합하여 하나의 릴레이션을 반환                    |
| 교집합 (INTERSECT, $\cap$ )                | - 이항연산, 2개의 릴레이션에서 중복된 내용을 선택하여 새로운 릴레이션을 반환            |
| 차집합 (DIFFERENCE, $-$ )                  | - 이항연산, 하나의 릴레이션에서 또 다른 릴레이션과 겹치는 내용을 제거하여 새로운 릴레이션을 반환 |
| 카티션 프로덕트 (CARTESIAN PRODUCT, $\times$ ) | - 이항연산, 2개의 릴레이션의 튜플들로 구성 가능한 모든 조합을 만들어 새로운 릴레이션을 반환   |





## 02. 관계형 데이터베이스와 관리구문

### SQL의 개요

#### 관계 연산자의 종류와 특징

##### ✓ 관계 연산자 - 순수 집합 연산자

| 연산자                        | 설명   |
|----------------------------|--|
| 선택 (SELECT, $\sigma$ )     | - 단항연산, 하나의 릴레이션에서 조건을 만족하는 튜플들을 선택하여 반환   |
| 프로젝션 (PROJECT, $\pi$ )     | - 단항연산, 하나의 릴레이션에서 조건을 만족하는 속성을 선택하여 반환  |
| 조인 (JOIN, $\bowtie$ )      | - 이항연산, 두개의 릴레이션에서 조건에 맞는 튜플 혹은 속성을 조합하여 새로운 릴레이션을 반환  |
| 디비전<br>(DIVISION, $\div$ ) | - 이항연산, $X \supset Y$ 인 2개의 릴레이션에서 R(X)와 S(Y)가 있을 때,<br>R의 속성이 S의 속성값을 모두 가진 튜플에서 S가 가진 속성을 제외한<br>속성만 분리하여 반환 |

## 02. 관계형 데이터베이스와 관리구문

### SQL의 개요

#### 관계 연산자의 종류와 특징

##### ✓ 관계 연산자의 예

| class-A |           |       |
|---------|-----------|-------|
| name    | course    | score |
| chris   | algorithm | 90    |
| tommy   | computer  | 90    |
| hans    | data      | 80    |
| harry   | db        | 90    |

| class-B |          |       |
|---------|----------|-------|
| name    | course   | score |
| chris   | db       | 80    |
| tommy   | computer | 90    |
| hans    | data     | 80    |
| harry   | OS       | 90    |



class-A  $\cap$  class-B

| class-A $\cap$ class-B |          |       |
|------------------------|----------|-------|
| name                   | course   | score |
| tommy                  | computer | 90    |
| hans                   | data     | 80    |

$\sigma$  score  $\geq$  90 (class-B)

| $\sigma$ score $\geq$ 90 (class-A) |          |       |
|------------------------------------|----------|-------|
| name                               | course   | score |
| tommy                              | computer | 90    |
| harry                              | OS       | 90    |

$\pi$  name, score (class-A)

| $\pi$ name, score (class-A) |       |
|-----------------------------|-------|
| name                        | score |
| chris                       | 90    |
| tommy                       | 90    |
| hans                        | 80    |
| harry                       | 90    |

## 02. 관계형 데이터베이스와 관리구문

### SQL의 개요

#### SQL 문의 종류

##### ✓ SQL 문의 종류

| SQL 문<br>종류     | 설명  |
|-----------------|---|
| DDL             | <ul style="list-style-type: none"><li>- Data Definition Language</li><li>- 데이터베이스의 논리적 구조를 정의하기 위한 언어, 데이터 사전에 저장</li><li>- 기본 Auto Commit</li></ul>          |
| DML             | <ul style="list-style-type: none"><li>- Data Manipulation Language</li><li>- 데이터베이스에 저장된 데이터를 조작 하기 위해 사용하는 언어</li><li>- 검색, 추가, 삭제, 갱신 작업 수행</li></ul>       |
| DCL<br>/<br>TCL | <ul style="list-style-type: none"><li>- Data Control Language</li><li>- Transaction Control Language</li><li>- 데이터에 대한 접근 권한 부여, 트랜잭션 등을 관리하기 위한 언어</li></ul> |

## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DML

#### DML의 세부 명령어

##### ✓ DML

- Data Manipulation Language
- 데이터베이스의 데이터를 조작, 검색/추가/삭제/갱신 등 수행

| 명령어    | 설명                                   |
|--------|--------------------------------------|
| INSERT | - 특정 테이블에 새로운 레코드를 삽입하는 명령어          |
| UPDATE | - 특정 테이블의 선택된 레코드의 속성값을 수정하는 명령어     |
| DELETE | - 특정 테이블의 선택된 레코드를 삭제하는 명령어          |
| SELECT | - 특정 테이블에서 조건에 맞는 레코드를 선택하여 반환하는 명령어 |

## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DDL

#### DDL의 세부 명령어

##### ✓ DDL

- Data Definition Language
- 데이터 베이스의 구조를 정의, 데이터 사전에 저장

| 명령어    | 설명                                |
|--------|-----------------------------------|
| CREATE | - 데이터베이스, 테이블, 인덱스, 뷰 등을 생성하는 명령어 |
| DROP   | - 테이블과 같은 데이터베이스 내 객체를 삭제하는 명령어   |
| ALTER  | - 데이터베이스 내 객체의 정의 혹은 속성을 변경하는 명령어 |
| RENAME | - 데이터베이스 내 객체의 이름을 변경하는 명령어       |

## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DDL

#### DDL의 세부 명령어 - CREATE

##### ✓ CREATE TABLE (1) - 테이블 생성

- 명령어 형식 : CREATE TABLE 테이블명 (컬럼명 데이터타입 제약조건, ...);
- 테이블 명, 컬럼명 규칙 : 알파벳, 숫자, \_ (언더바), \$ (달러), # (샵), 대소문자 구분하지 않음, 테이블명은 단수형 권장.
- 데이터 타입
  - CHAR(L) : 고정길이 문자열, 할당된 값의 길이나 L 보다 작을 경우 공백으로 채워짐
  - VARCHAR2(L), VARCHAR(L) : 가변길이 문자열, 최대 L의 길이의 값을 저장
  - NUMBER(L,D) : 숫자형, L (전체 자리 수), D (소수점 자리 수)
  - DATE, DATETIME : 날짜형

## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DDL

#### DDL의 세부 명령어 - CREATE

##### ✓ CREATE TABLE (2)

<참고>

ALTER TABLE 테이블명 ADD CONSTRAINT 칼럼명 FOREIGN KEY (칼럼명) REFERENCES 테이블명(칼럼명);

##### • 제약사항

- PRIMARY KEY : 기본키 정의 (기본키 생성시 DBMS가 자동으로 인덱스를 생성함, NULL 불가)
- FOREIGN KEY : 다른 테이블의 기본키를 외래키로 지정 (참조 무결성 제약조건)
- UNIQUE KEY : 고유한 속성 값 (NULL 가능)
- NOT NULL : NULL이 될 수 없음
  - ※ NULL: 아직 정의되지 않은 값 또는 현재 데이터를 입력하지 못하는 값을 의미
- CHECK : 입력 값의 종류 및 범위 제한 확인
- DEFAULT : 데이터 입력 시, 지정 값이 없을 경우 기본값 사용



## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DDL

#### DDL의 세부 명령어 - CREATE

##### ✓ CREATE TABLE 의 예

```
CREATE TABLE employees (  
  employee_id INT NOT NULL,  
  first_name VARCHAR(50),  
  last_name VARCHAR(50) NOT NULL,  
  email VARCHAR(100) NOT NULL,  
  job_id VARCHAR(50) NOT NULL,  
  hire_date DATE NOT NULL,  
  PRIMARY KEY (employee_id),  
  FOREIGN KEY (job_id) REFERENCES jobs(job_id)  
);
```

```
ALTER TABLE employees  
ADD salary DECIMAL(8, 2);
```

```
ALTER TABLE employees  
ADD CONSTRAINT email_unique UNIQUE (email);
```

```
ALTER TABLE employees  
DROP CONSTRAINT email_unique;
```

```
ALTER TABLE employees  
DROP COLUMN email;
```



## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DDL

#### DDL의 세부 명령어 - DROP, TRUNCATE

##### ✓ DROP TABLE - 테이블 제거

- 명령어 형식 : DROP TABLE 테이블명;
- 테이블의 데이터와 구조 모두를 삭제, 복구 불가

##### ✓ TRUNCATE TABLE - 테이블 (데이터) 제거

- 명령어 형식 : TRUNCATE TABLE 테이블명;
- 테이블의 전체 데이터(만) 삭제, ROLLBACK 불가 (로그를 기록하지 않음)

<참고>

- CASCADE CONSTRAINT : 해당 테이블과 관계가 있는  
참조 제약조건도 함께 삭제

## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DDL

#### DDL의 세부 명령어 - ALTER

##### ✓ ALTER TABLE - 테이블 수정

- 명령어 형식 :

- ALTER TABLE 테이블명 ADD (컬럼명, 데이터타입);
- ALTER TABLE 테이블명 DROP COLUMN 컬럼명;
- ALTER TABLE 테이블명 MODIFY (컬럼명, 데이터타입, 제약조건)

- MODIFY 시 제약사항

- 칼럼의 크기 축소 시 NULL 혹은 값이 없는 경우에만 가능
- 데이터 유형 변경 시 NULL만 있는 경우에 가능
- NOT NULL 설정은 NULL이 없을 경우에 가능

<참고>

- DROP CONSTRAINT : 제약 조건 삭제
- ADD CONSTRAINT : 제약 조건 추가

## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DDL

#### DDL의 세부 명령어 - RENAME, DESCRIBE

- ✓ RENAME TABLE - 이름 변경
  - 명령어 형식 :
    - RENAME TABLE 테이블명 TO 새로운 테이블명
- ✓ DESCRIBE TABLE - 테이블 구조 확인
  - 명령어 형식 :
    - DESCRIBE 테이블명
  - ※ SQL Sever : sp\_help dbo.테이블명

## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - TCL

#### TCL의 세부 명령어

##### ✓ TCL

- Transaction Control Language
- 논리적인 작업 단위의 실행 결과를 트랜잭션 별로 제어하는 명령어

| 명령어      | 설명                         |
|----------|----------------------------|
| COMMIT   | - 수행한 작업은 데이터베이스에 반영       |
| ROLLBACK | - 수행한 작업을 취소하고 수행 전 상태로 복귀 |

## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DCL

#### DCL의 세부 명령어

##### ✓ DCL

- Data Control Language
- 데이터에 대한 접근 권한 부여 등 관리 실행

| 명령어    | 설명                             |
|--------|--------------------------------|
| GRANT  | - 데이터베이스 사용자에게 특정 권한을 부여하는 명령어 |
| REVOKE | - 데이터베이스 사용자의 특정 권한을 제거하는 명령어  |

## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DCL

#### DCL의 세부 명령어 - USER 관리

##### ✓ CREATE / ALTER / DROP USER

- 사용자 생성

- CREATE USER 사용자명 IDENTIFIED BY 패스워드;

- 사용자 변경

- ALTER USER 사용자명 IDENTIFIED BY 패스워드;

- 사용자 삭제

- DROP UESER 사용자명;

<참고 : 오라클의 유저>

- SCOTT : 테스트용 샘플 유저
- SYS : DBA 권이 부여된 관리자 유저
- SYSTEM : DB의 모든 시스템 권한이 부여된 DBA

## 02. 관계형 데이터베이스와 관리구문

### 관리구문 - DCL

#### DCL의 세부 명령어 - GRANT

##### ✓ GRANT - 권한 부여

- 명령어 형식 :

- GRANT 권한 ON 대상 TO 사용자;

GRANT SELECT, INSERT, UPDATE ON employees TO user\_john;

GRANT ALL PRIVILEGES ON employees TO user\_john;

- 권한의 종류

- DML 관련 : SELECT, INSERT, UPDATE, DELETE, ALTER, ALL

- 제약조건 관련 : REFERENCES

- 인덱스 관련 : INDEX

- CREATE ROLE 롤이름 형식으로 ROLE을 생성 후 사용할 수 있음.

<참고 : 오라클의 ROLE>

- CONNECT : CREATE SESSION 권한

- RESOURCE : CREATE CLUSTER/TABLE/OPERATOR 등의 권한

# SQLD

## 자격증 특강

컴퓨터·AI공학과  
김윤수 교수



## 03. 데이터 모델과 SQL

### 데이터 모델과 SQL

- ✓ 정규화
- ✓ 관계와 조인의 이해
- ✓ 트랜잭션의 이해
- ✓ Null 속성의 이해
- ✓ 본질식별자 vs 인조식별자

## 03. 데이터 모델과 SQL

### 정규화

#### 함수적 종속성의 개념과 기본 원리

##### ✓ 함수적 종속성의 개념

- 데이터들이 상호 종속되는 현상, 결정자와 종속자의 관계
- $X \rightarrow Y$ : 결정자인 X가 종속자인 Y를 결정

##### ✓ 함수적 종속성의 기본 원리

- 속성의 부분집합  $X \rightarrow Y$ 에서 Y에 대응되는 X는 하나만 존재
- X가 기본키인 경우, 모든 속성 Y는 X에 종속
- $X \rightarrow Y$ 인 관계에서 X가 후보키가 아닐 경우 이상현상이 발생할 수 있음

▶ 이상현상 : 릴레이션에 데이터를 조작할 때 의도하지 않는 현상이 발생하거나 데이터 불일치 혹은 누락이 발생하는 현상으로 '삽입이상, 갱신이상, 삭제이상'이 있음

## 03. 데이터 모델과 SQL

### 정규화

#### 함수적 종속성의 종류

##### ✓ 함수적 종속성의 종류

| 종류       | 설명   |
|----------|--|
| 완전함수 종속성 | - $X, Y \rightarrow Z$ 일때, $X \rightarrow Z$ 가 아니고 $Y \rightarrow Z$ 가 아니면 $Z$ 는 완전함수 종속 |
| 이행함수 종속성 | - $X \rightarrow Y$ 이고 $Y \rightarrow Z$ 일때, $X \rightarrow Z$ 를 만족하는 경우                 |
| 부분함수 종속성 | - $X, Y \rightarrow Z$ 일때, $X \rightarrow Z$ 이거나 $Y \rightarrow Z$ 인 경우                  |

## 03. 데이터 모델과 SQL

### 정규화

#### 정규화의 개념과 특징

##### ✓ 정규화(Normalization)의 개념

- 불필요한 데이터 중복에 따른 이상현상 제거를 위해 무손실 분해하는 과정
- 함수적 종속성을 이용하여 릴레이션이 연관성 있는 속성 들로만 구성되도록 분해하는 과정

※ 정규형 : 정규화로 도출된 데이터 모델이 갖추어야 할 특성

##### ✓ 정규화의 특징

- 원칙 : 무손실 분해, 함수적 종속성, 최소의 데이터 중복성, 분리의 원칙
- 장점 : 이상현상 제거와 일관성 유지, 중복제거 통한 저장공간 절약, 데이터 구조의 유연성 확보
- 단점 : Join을 통한 정보 구성 시 성능 저하

## 03. 데이터 모델과 SQL

### 정규화

#### 정규화의 유형

##### ✓ 정규화의 유형

정규화 수행 이후, 성능향상과 운영 단순화를 위해 **반 정규화**를 고려한다.  
- 모델 통합 (테이블 반정규화), 부가 정보 추가 (컬럼 반정규화, 중복관계 추가)

| 정규화의 유형 | 설명  |
|---------|---|
| 1차 정규화  | <ul style="list-style-type: none"><li>- 속성의 원자값 확보</li><li>- 반복되는 속성 제거, 다중값 속성의 분리</li><li>- R에 속한 모든 도메인이 원자값(atomic value)만으로 구성</li></ul>   |
| 2차 정규화  | <ul style="list-style-type: none"><li>- 부분함수 종속성 제거</li><li>- 릴레이션 R이 1NF이고, 릴레이션의 기본키가 아닌 속성들이 기본키에 완전히 함수적으로 종속</li><li>- 기본키가 하나의 컬럼일 경우 생략 가능</li></ul>                           |
| 3차 정규화  | <ul style="list-style-type: none"><li>- 이행함수 종속성 제거</li><li>- 릴레이션 R이 2NF이고, 기본키가 아닌 모든 속성들이 기본키에 대하여 이행적 함수 종속성의 관계를 가지지 않음.</li><li>- 즉, 기본키 외의 속성들 간에 함수적 종속성을 가지지 않는 경우</li></ul> |
| 기타      | <ul style="list-style-type: none"><li>- BCNF, 4차 정규화, 5차 정규화</li></ul>  |

## 03. 데이터 모델과 SQL

### 관계와 조인의 이해

#### 정규화, 조인, 반정규화

##### ✓ 정규화와 성능

- 정규화의 장점 : 데이터 이상현상 제거, 데이터 중복 최소화, 입력/수정/삭제 시 성능 향상
- 정규화의 단점(고려사항) : 조회 시 처리 조건에 따라 성능 저하 발생

➤ 조인 (JOIN) : 둘 이상의 테이블을 연결하여 논리적 관계를 기준으로 검색하여 결과 집합을 만드는 기능(과정)

##### ✓ 반정규화 - 데이터 중복을 허용하여 조회 성능 향상, 정규화 이후 수행

- 반정규화 절차 : 대상 조사 → 대안 검토 (뷰, 클러스터링, 인덱스, 어플리케이션 등) → 반정규화 적용

##### • 반정규화 기법

➤ 테이블 반정규화 : 테이블 병합, 테이블 분할, 테이블 추가

➤ 컬럼 반정규화 : 중복 컬럼 추가, 파생 컬럼 추가, 이력 테이블 컬럼 추가

<참고>

- 관계 반정규화 : 중복 관계 추가

## 03. 데이터 모델과 SQL

### 트랜잭션의 이해

#### 트랜잭션(Transaction)의 개념과 특징

##### ✓ 트랜잭션의 개념

- 데이터베이스의 무결성을 보장하며 요청된 작업을 완료하기 위하여, 한번에 수행되어야 할 일련의 Read와 Write연산을 정의한 논리적 작업 단위
- 분할할 수 없는 최소단위, All or Nothing

##### ✓ 트랜잭션의 특징

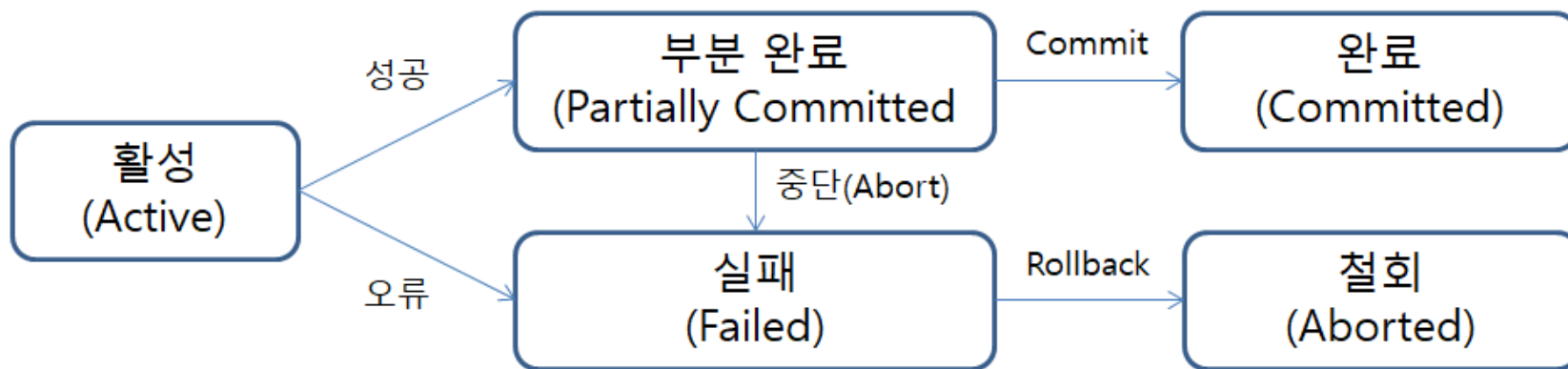
- 원자성(Atomicity) : All or Nothing, 완전히 완료되지 않았을 수행되기 전과 같아야 함.
- 일관성(Consistency) : 실행이 성공적으로 완료되면 데이터베이스는 모순없이 일관성이 보존된 상태
- 고립성(Isolation) : 트랜잭션의 부분적 실행결과를 다른 트랜잭션이 접근할 수 없음
- 지속성(Durability) : 완료된 트랜잭션의 결과는 영구적으로 데이터베이스에 저장

## 03. 데이터 모델과 SQL

### 트랜잭션의 이해

#### 트랜잭션(Transaction)의 상태 다이어그램

##### ✓ 트랜잭션의 상태 다이어그램



<출처: <https://itwiki.kr/w/%ED%8A%B8%EB%9E%9C%EC%9E%AD%EC%85%98> >



## 03. 데이터 모델과 SQL

### 트랜잭션의 이해

#### 트랜잭션(Transaction) 상태와 관련 명령어

##### ✓ 트랜잭션의 상태 다이어그램

- 실행(Active) : 트랜잭션 실행 시작 혹은 실행 중인 상태 (START TRANSACTION)
- 부분완료(Partially Committed) : 트랜잭션의 마지막 명령을 실행한 후의 상태
- 실패(Failed) : 트랜잭션을 더 이상 정상적으로 실행할 수 없음을 발견한 상태
- 철회(Abort) : 트랜잭션이 실패하여 취소되어 시작 전 상태로 환원된 상태 (ROLLBACK)
- 완료(Committed) : 트랜잭션이 성공적으로 완료된 상태 (COMMIT)

※ SAVEPOINT : 롤백 기준점 설정

| 명령어      | 설명                         |
|----------|----------------------------|
| COMMIT   | - 수행한 작업은 데이터베이스에 반영       |
| ROLLBACK | - 수행한 작업을 취소하고 수행 전 상태로 복귀 |

## 03. 데이터 모델과 SQL

### 트랜잭션의 이해

#### 트랜잭션(Transaction) 상태와 관련 명령어

##### ✓ 트랜잭션의 관련 명령어의 예

```
BEGIN TRANSACTION; -- 트랜잭션 시작
```

```
SAVEPOINT savepoint1; -- 세이브포인트 설정
```

```
UPDATE employees SET last_name = 'Smith' WHERE employee_id = 207;  
-- 직원의 성을 업데이트
```

```
ROLLBACK TO savepoint1; -- 세이브포인트로 롤백
```

```
COMMIT; -- 변경 사항을 데이터베이스에 영구적으로 저장
```

## 03. 데이터 모델과 SQL

### Null 속성의 이해

#### NULL의 개념과 특징

##### ✓ NULL의 개념

- 값이 없는 상태, 즉 존재하지 않는 상태를 의미
- 구체적인 값이 입력되지 않은 상태

##### ✓ NULL의 특징

- 동일한 ROW (레코드)에 대한 연산의 결과값은 NULL이 됨.
- 다른 인스턴스의 데이터와 함께 연산할 경우 NULL 값은 제외됨.

## 03. 데이터 모델과 SQL

### 본질식별자 vs 인조식별자

#### 본질식별자와 인조식별자의 개념

##### ✓ 본질 식별자와 인조 식별자의 개념

※ 비식별 관계가 된다.

| 본질 식별자   | 인조 식별자   |
|--|--|
| 업무, 업무 프로세스에 실제 존재하는 식별자<br>별도의 가공을 거치지 않은 원래의 식별자 | 업무적으로 생성된 것은 아니나<br>본질 식별자가 복잡한(2개 이상) 구성을 갖고 있으므로<br>인위적으로 만든 식별자 (대리식별자) |

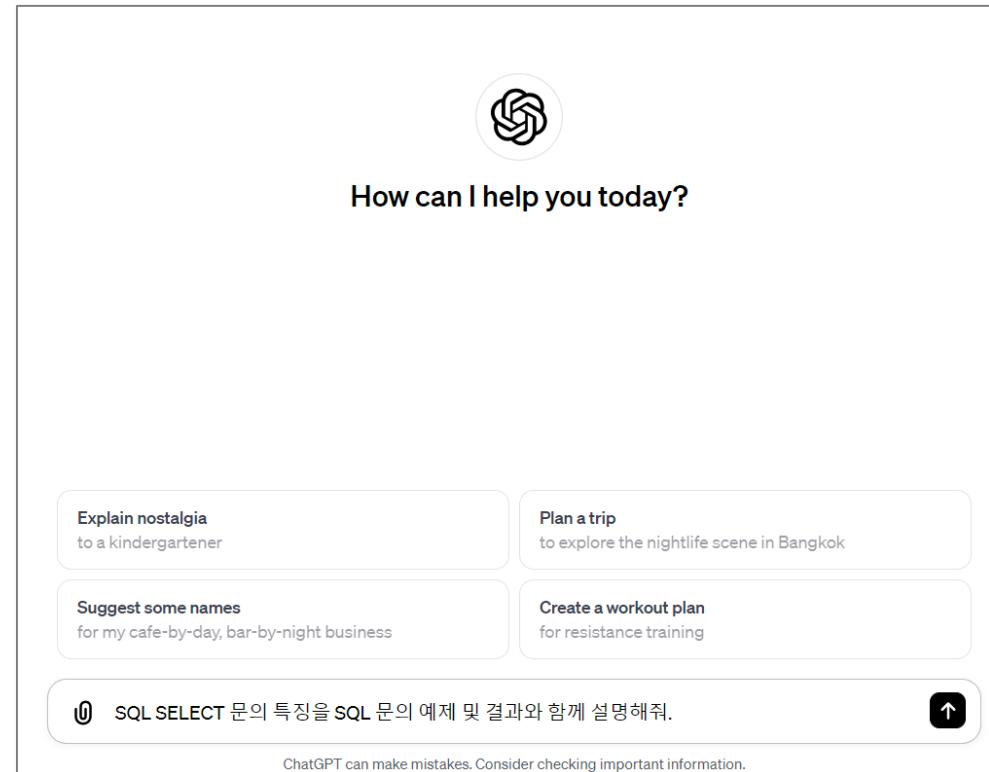


## 03. 데이터 모델과 SQL

### 〈추가〉 SQL을 학습, 실습을 위한 사이트

#### ChatGPT

✓ <https://chat.openai.com/>



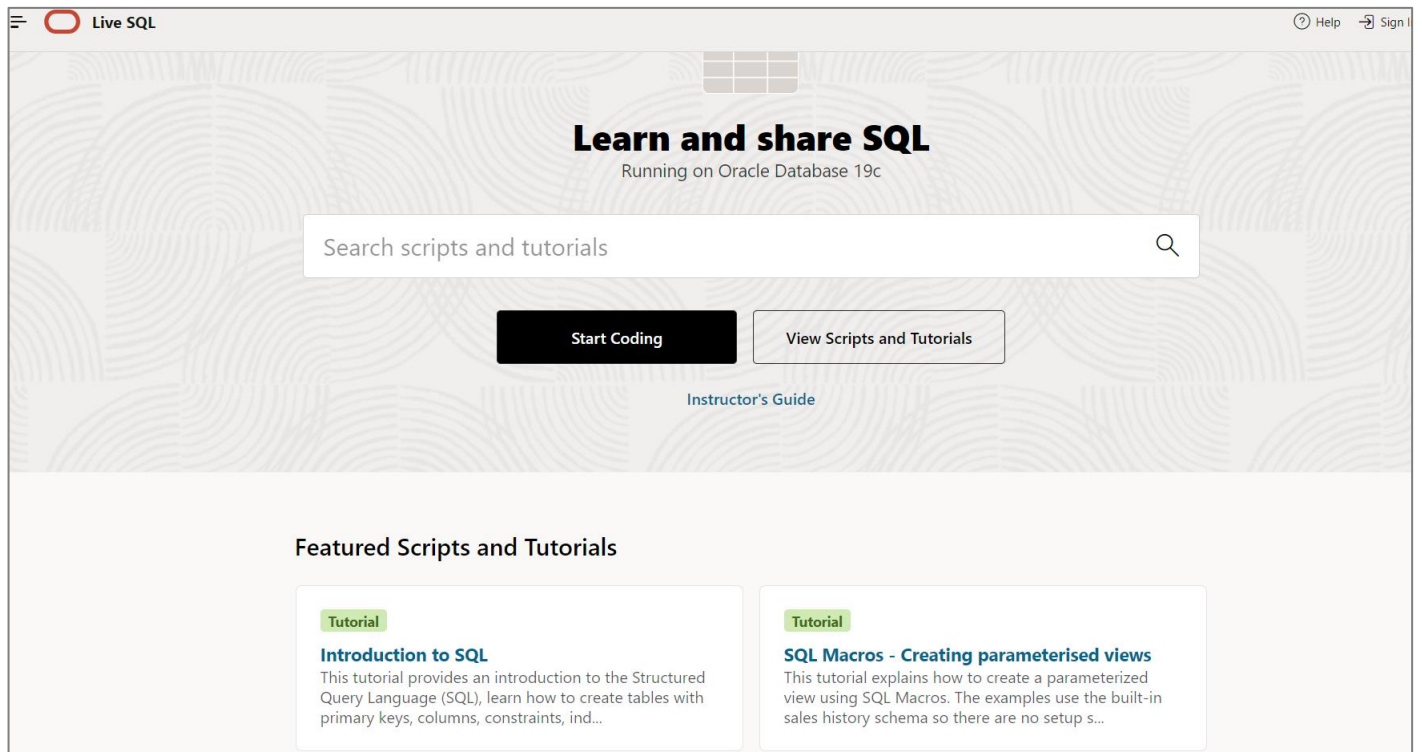


## 03. 데이터 모델과 SQL

〈추가〉 SQL을 학습, 실습을 위한 사이트

### ORACLE - Live SQL

✓ <https://livesql.oracle.com>





## 03. 데이터 모델과 SQL

### 〈추가〉 SQL을 학습, 실습을 위한 사이트

#### W3 Schools

✓ <https://www.w3schools.com/sql/>

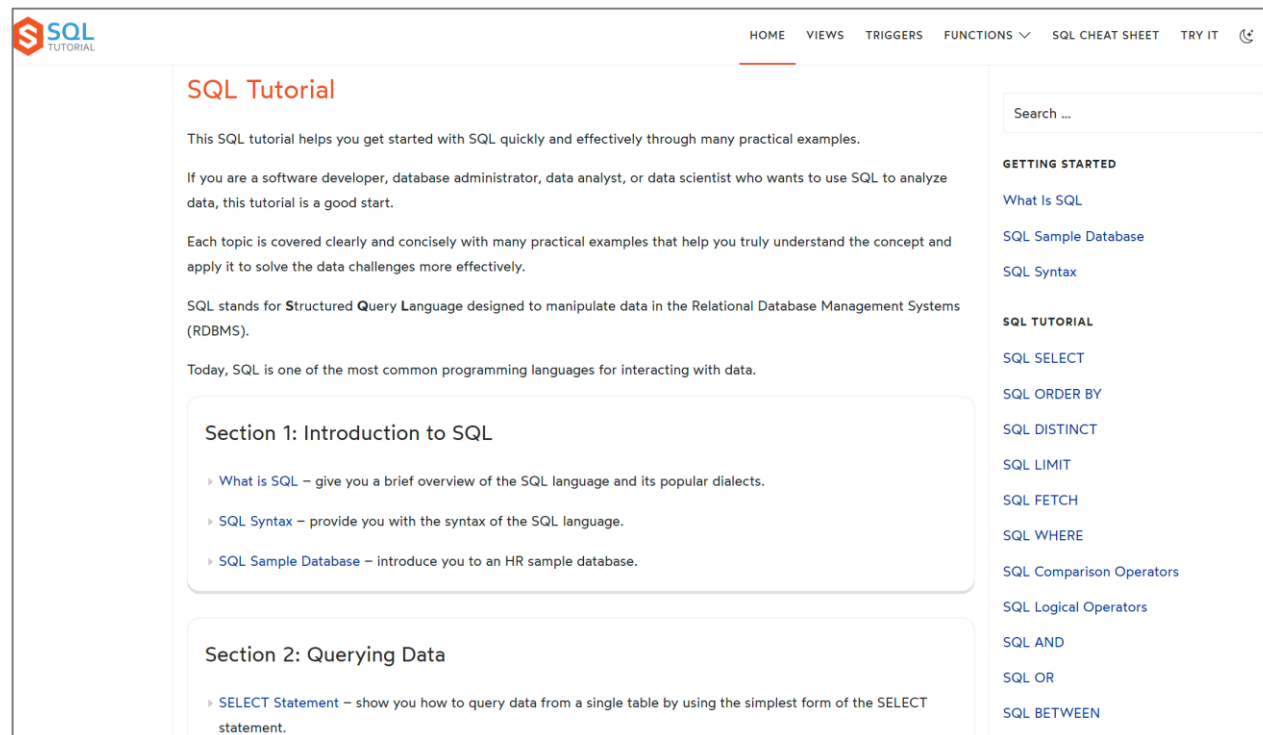
The screenshot shows the W3 Schools website's SQL Tutorial page. The top navigation bar includes links for Tutorials, Exercises, Certificates, and Services, along with a search bar and buttons for Sign Up and Log in. A secondary navigation bar lists various technologies, with 'SQL' highlighted. On the left, a sidebar menu lists SQL topics, with 'SQL HOME' selected. The main content area features the title 'SQL Tutorial', navigation buttons for '< Home' and 'Next >', and introductory text about SQL as a standard language for databases. A prominent green button invites users to 'Start learning SQL now ». On the right, there are promotional banners for 'Modern Web Development' and a 'COLOR PICKER' tool.

## 03. 데이터 모델과 SQL

〈추가〉 SQL을 학습, 실습을 위한 사이트

### SQL Tutorial

✓ <https://www.sqltutorial.org/>



The screenshot shows the homepage of the SQL Tutorial website. The header includes a navigation menu with links for HOME, VIEWS, TRIGGERS, FUNCTIONS, SQL CHEAT SHEET, and TRY IT. The main content area is titled "SQL Tutorial" and contains an introductory paragraph about the tutorial's purpose. Below this, there are two sections: "Section 1: Introduction to SQL" and "Section 2: Querying Data". Each section lists several topics with brief descriptions. A right-hand sidebar contains a search bar and a list of links under the heading "GETTING STARTED", including "What Is SQL", "SQL Sample Database", "SQL Syntax", and a list of SQL keywords and operators.

**SQL Tutorial**

This SQL tutorial helps you get started with SQL quickly and effectively through many practical examples.

If you are a software developer, database administrator, data analyst, or data scientist who wants to use SQL to analyze data, this tutorial is a good start.

Each topic is covered clearly and concisely with many practical examples that help you truly understand the concept and apply it to solve the data challenges more effectively.

SQL stands for **Structured Query Language** designed to manipulate data in the Relational Database Management Systems (RDBMS).

Today, SQL is one of the most common programming languages for interacting with data.

**Section 1: Introduction to SQL**

- › **What is SQL** – give you a brief overview of the SQL language and its popular dialects.
- › **SQL Syntax** – provide you with the syntax of the SQL language.
- › **SQL Sample Database** – introduce you to an HR sample database.

**Section 2: Querying Data**

- › **SELECT Statement** – show you how to query data from a single table by using the simplest form of the SELECT statement.

**GETTING STARTED**

- [What Is SQL](#)
- [SQL Sample Database](#)
- [SQL Syntax](#)
- SQL TUTORIAL**
- [SQL SELECT](#)
- [SQL ORDER BY](#)
- [SQL DISTINCT](#)
- [SQL LIMIT](#)
- [SQL FETCH](#)
- [SQL WHERE](#)
- [SQL Comparison Operators](#)
- [SQL Logical Operators](#)
- [SQL AND](#)
- [SQL OR](#)
- [SQL BETWEEN](#)





## 03. 데이터 모델과 SQL

### 〈추가〉 SQL을 학습, 실습을 위한 사이트

sqlzoo.net

✓ [https://sqlzoo.net/wiki/SQL\\_Tutorial](https://sqlzoo.net/wiki/SQL_Tutorial)

The screenshot displays the SQLZoo website interface. At the top, there's a navigation bar with the SQLZoo logo and a link to 'DataWars: Practice Data Science/Analysis with +100 Real Life Projects'. The main heading is 'SQL Tutorial'. Below it, a sub-heading reads 'Tutorials: Learn SQL step by step'. The content is organized into a list of topics, each with a brief description and a link to the next step. The topics include: 0 SELECT basics, 1 SELECT name, 2 SELECT from World, 3 SELECT from Nobel, 4 SELECT within SELECT, 5 SUM and COUNT, 6 JOIN, 7 More JOIN operations, 8 Using Null, 8+ Numeric Examples, 9- Window function, and 9+ COVID 19. On the left side, there's a sidebar with a 'Save Progress' section and a 'Reference' section. The 'Save Progress' section lists various SQL topics with progress indicators (bars). The 'Reference' section includes links to 'NoSQL zoo', 'SELECT Functions', 'SELECT .. WHERE', 'SELECT .. GROUP BY', 'SELECT .. JOIN', 'SELECT .. SELECT', 'INSERT .. VALUES', and 'INSERT .. SELECT'.

**SQLZoo**

**DataWars:** Practice Data Science/Analysis with +100 Real Life Projects

### SQL Tutorial

Tutorials: Learn SQL step by step

- 0 SELECT basics**  
Some simple queries to get you started
- 1 SELECT name**  
Some pattern matching queries
- 2 SELECT from World**  
In which we query the World country profile table.
- 3 SELECT from Nobel**  
Additional practice of the basic features using a table of Nobel Prize winners.
- 4 SELECT within SELECT**  
In which we form queries using other queries.
- 5 SUM and COUNT**  
In which we apply aggregate functions. [more the same](#)
- 6 JOIN**  
In which we join two tables; game and goals. [previously music tutorial](#)
- 7 More JOIN operations**  
In which we join actors to movies in the Movie Database.
- 8 Using Null**  
In which we look at teachers in departments. [previously Scottish Parliament](#)
- 8+ Numeric Examples**  
In which we look at a survey and deal with some more complex calculations.
- 9- Window function**  
In which we examine UK general election results.
- 9+ COVID 19**  
In which we measure the impact of COVID-19

**Save Progress**

- SELECT basics
- quiz
- SELECT from world
- quiz
- SELECT from nobel
- quiz
- SELECT in SELECT
- quiz
- SUM and COUNT
- quiz
- JOIN
- quiz
- More JOIN
- quiz
- Using NULL
- quiz
- Self JOIN
- quiz

**Reference**

- NoSQL zoo
- SELECT Functions
- SELECT .. WHERE
- SELECT .. GROUP BY
- SELECT .. JOIN
- SELECT .. SELECT
- INSERT .. VALUES
- INSERT .. SELECT



## 03. 데이터 모델과 SQL

### 〈추가〉 SQL을 학습, 실습을 위한 사이트

#### SQL Teaching

✓ <https://www.sqlteaching.com/>

### SQL Teaching

**SELECT \***

SELECT specific columns

WHERE ... Equals

WHERE ... Greater than

WHERE ... Greater than or equal

AND

OR

IN

DISTINCT

ORDER BY

LIMIT # of returned rows

COUNT(\*)

COUNT(\*) WHERE

#### Lesson 1: SELECT \*

In SQL, data is usually organized in various tables. For example, a sports team database might have the tables *teams*, *players*, and *games*. A wedding database might have tables *guests*, *vendors*, and *music\_playlist*.

Imagine we have a table that stores family members with each member's name, species, gender, and number of books read.

Let's start by grabbing all of the data in one table. We have a table called **family\_members** that is shown below. In order to grab all of that data, please run the following command: `SELECT * FROM family_members;`

The `*` above means that all of the columns will be returned, which in this case are *id*, *name*, *gender*, *species*, and *num\_books\_read*.

Note: This tutorial uses the [SQLite](#) database engine. The different variants of SQL use slightly different syntax.

Stuck? Email us at [sqlteaching@gmail.com](mailto:sqlteaching@gmail.com) for help or feedback. [View on GitHub](#)

Redesigned by [Alvaro Cervan](#) 2023

# SQLD

## 자격증 특강

컴퓨터·AI공학과  
김윤수 교수



# 04. SQL 기본

## SQL 기본

- ✓ SELECT 문
- ✓ 함수
- ✓ WHERE 절
- ✓ GROUP BY, HAVING 절
- ✓ ORDER BY 절
- ✓ 조인

# 04. SQL 기본

## SQL의 시작

### SQL의 시작 - INSERT, UPDATE, DELETE

#### ✓ INSERT - 데이터 입력

- 명령어 형식 :

- INSERT INTO 테이블명 (컬럼명, ...) VALUES (값, ...);
- INSERT INTO 테이블명 VALUES (값, ...);

```
INSERT INTO employees (employee_id, first_name, last_name, email, hire_date, job_id)
VALUES (207, 'Laura', 'Bissot', 'LBISSOT', TO_DATE('2024-01-21', 'YYYY-MM-DD'), 'IT_PROG');
```

#### ✓ UPDATE - 데이터 수정

- 명령어 형식 :

- UPDATE 테이블명 SET 컬럼명=값;

```
UPDATE employees
SET job_id = 'AC_ACCOUNT'
WHERE employee_id = 207;
```

# 04. SQL 기본

## SQL의 시작

### SQL의 시작 - INSERT, UPDATE, DELETE

#### ✓ DELETE - 데이터 삭제

- 명령어 형식 :

- DELETE FROM 테이블명 WHERE 조건절;

- DELETE FROM 테이블명;

※ 데이터를 삭제해도 테이블 용량을 초기화 하지 않음

```
DELETE FROM employees WHERE employee_id = 207;
```

# 04. SQL 기본

## SELECT 문

### SELECT 문의 기본

#### ✓ SELECT - 데이터 선택/가져오기

- 명령어 형식 :

- SELECT 컬럼명 FROM 테이블명;

- SELECT DISTINCT 컬럼명 FROM 테이블명;

- SELECT \* FROM 테이블명;

※ DISTINCT (중복제거)

※ ALIAS (별칭)을 설정하여 간결하고 명확한 SQL 작성 가능, 별칭을 사용한 경우 별칭을 이용한 참조 필수

<참고>

- 문자열 합성 연산자 : '+', CONCAT, '||' (오라클)

- DUAL : 오라클이 제공하는 기본 더미 테이블, 연산 수행위해 사용

```
SELECT S.NAME, B.NAME FROM student_t S, band_t B WHERE S.bcode = B.bcode;
```

# 04. SQL 기본

## SELECT 문

### SELECT 문 - 연산자의 활용

- ✓ SELECT - 산술연산자 활용
  - 산술 연산자 : 수학의 사칙연산, NUMBER / DATE 등의 데이터에 사용
    - 괄호 ( ) : 가장 먼저 처리해야 할 것 지정
    - \*, /, +, - : 사칙연산
    - % : 나머지 연산 (SQL Server)

```
SELECT 1+1, 10/2 FROM DUAL;
```

```
SELECT C1+C2 AS A, C1-C2 AS B FROM sample_t;
```



# 04. SQL 기본

## SELECT 문

### SELECT 문 - 연산자의 활용

#### ✓ SELECT - 문자 연결 연산자

- 문자 연결 연산자 : 여러 개의 문자열을 하나의 문자열로 합성

➤ || (오라클), + (SQL 서버)

```
SELECT 'H' || 'I' AS HI from DUAL;
```

```
SELECT NAME || '선수, HIGHT || 'cm' FROM member_t;
```

# 04. SQL 기본

## 함수

### 문자 함수의 종류와 특징

#### ✓ 문자함수의 종류와 특징 (1)

- CHR (숫자) : 아스키 코드에 대한 문자 출력
- LOWER (문자열) : 소문자열로 변환 , UPPER (문자열) : 대문자열로 변환
- LTRM (문자열, 점검문자) : 문자열의 왼쪽부터 점검문자를 제거, 점검문자를 지정하지 않으면 공백 문자를 대상으로 함
- RTRM (문자열, 점검문자) : 문자열의 오른쪽부터 점검문자를 제거, 점검문자를 지정하지 않으면 공백 문자를 대상으로 함
- TRM (문자열, 점검문자) : 문자열의 양쪽으로부터 점검문자를 제거, 점검문자를 지정하지 않으면 공백 문자를 대상으로 함

```
SELECT LTRIM('    CHRIS') FROM DUAL;
```

```
SELECT TRM('~~~Hi~~~', '~') FROM DUAL;
```

# 04. SQL 기본

## 함수

### 문자 함수의 종류와 특징

#### ✓ 문자함수의 종류와 특징 (2)

- SUBSTR (문자열, 시작점, 길이) : 문자열의 시작위치 부터 길이(개수)만큼 추출, 길이를 지정하지 않은 경우 끝까지 추출
- LENGTH (문자열) : 문자열의 길이 반환
- REPLACE (문자열, 대상 문자열, 대체 문자열) : 문자열에 포함된 대상 문자열을 대체, 대체 문자열을 지정하지 않으면 제거
- LPAD (문자열, 길이, 문자) : 설정한 길이의 문자열이 될 때까지 문자열의 외쪽을 지정한 문자로 채움

```
SELECT SUBSTR('MyHome', 3) FROM DUAL;
```

```
SELECT SUBSTR('YourHome', 1, 3) FROM DUAL;
```

```
SELECT REPLACE('MyHome', 'My', 'Your') FROM DUAL;;
```

```
SELECT LPAD('Hi', 5, '~') FROM DUAL;
```

# 04. SQL 기본

## 함수

### 숫자 함수의 종류와 특징

#### ✓ 숫자 함수의 종류와 특징 (1)

- ABS (숫자) : 절대값을 반환
- SIGN (숫자) : 수의 부호를 반환 (양수 : 1, 음수 : -1, 0 : 0)
- ROUND (숫자, 자리수) : 지정된 소수점 자리수로 반올림, 자리수를 지정하지 않을 경우 0을 기본값을 사용
- TRUNC (숫자, 자리수) : 지정된 소수점 자리수까지 버림하여 반환, 자리수를 지정하지 않을 경우 0을 기본값으로 사용

```
SELECT ROUND(123.76, 1) FROM DUAL;
```

```
SELECT ROUND(173.76, -2) FROM DUAL;
```

```
SELECT TRUNC(123.76, 1) FROM DUAL;
```

```
SELECT TRUNC(123.76, -1) FROM DUAL;
```

# 04. SQL 기본

## 함수

### 숫자 함수의 종류와 특징

#### ✓ 숫자 함수의 종류와 특징 (2)

- CEIL (숫자) : 소수점 이하의 수를 올림 한 정수를 반환
- FLOOR(숫자) : 소수점 이하의 수를 버림 한 정수를 반환
- MOD(숫자1, 숫자2) : 나머지를 반환, 숫자2가 0일 경우 숫자1을 반환

```
SELECT CEIL(12.77) FROM DUAL;
```

```
SELECT FLOOR(25.4) FROM DUAL;
```

```
SELECT FLOOR(-25.4) FROM DUAL;
```

```
SELECT MOD(15, -2) FROM DUAL
```

```
SELECT MOD(-15, -2) FROM DUAL;
```

## 04. SQL 기본

### 함수

#### 날짜 함수의 종류와 특징

##### ✓ 날짜 함수의 종류와 특징

- SYSDATE : 현재의 연, 월, 시, 분, 초를 반환
- EXTRACT (단위 FROM 날짜) : 날짜 중 특정 단위의 데이터를 추출하여 반환  
(YEAR, MONTH, DAY, HOUR, MINUTE, SECOND)
- ADD\_MONTH(날짜, 개월 수) : 날짜에 지정한 개월 수를 더한 날짜를 반환 (기준 날짜가 없을 시 마지막 일자 반환)

```
SELECT SYSDATE FROM DUAL;
```

```
EXTRACT (YEAR FROM SYSDATE) AS YEAR FROM DUAL;
```

```
ADD_MONTH(TO_DATE('2024-01-10', 'YYYY-MM-DD'), 1) AS NEXT FROM DUAL;
```

# 04. SQL 기본

## 함수

### 변환 함수의 종류와 특징

#### ✓ 데이터 변환 방식의 종류

- 명시적 변환 : 데이터 변환 함수를 이용하여 명시적으로 변환
- 암시적 변환 : 연산 방식에 따라 자동으로 (내부적으로) 변환

#### ✓ 데이터 형 변환 함수

- TO\_NUMBER(문자열) : 숫자형으로 변환
- TO\_CHAR(데이터, 포맷) : 숫자, 날짜 데이터를 지정한 포맷의 문자열로 변환, 포맷 생략 시 전체를 문자열로 변환
- TO\_DATE(문자열, 포맷) : 포맷에 따라 날짜형으로 변환 (YYYY, MM, DD, HH, HH24, MI, SS)

```
SELECT TO_CHAR(123) FROM DUAL;
```

```
SELECT TO_CHAR(SYSDATE, 'YYYYMMDD HH24MISS') FROM DUAL;
```

## 04. SQL 기본

### 함수

#### NULL 관련 함수의 종류와 특징

##### ✓ NULL 관련 함수의 종류와 특징

- NVL(인자1, 인자2) : 인자 1이 NULL일 경우 인자2를 반환, NULL이 아닌 경우 인자 1을 반환
- NULLIF(인자1, 인자2) : 인자1과 인자2가 같으면 NULL을 반환, 같지 않은 경우 인자1을 반환
- COALESCE(인자1, 인자2, 인자3, ...) : 전달한 인자 중 NULL이 아닌 최초의 인자의 값을 반환
- NVL2(인자1, 인자2, 인자3) : 인자1이 NULL이 아닌 경우 인자2를 반환, NULL일 경우 인자3을 반환

```
SELECT USER_NO, NVL(GRADE, 0) AS HAKJUM FROM user_t;
```

```
SELECT USER_NAME, COALESCE (TEL, MAIL, SOCIAL) AS USER_CONTACT FROM user_t;
```

```
SELECT USER_NAME, NVL2(GRADE, '학점취득', '학점미취득') AS GRADE_CHECK FROM user_t;
```



# 04. SQL 기본

## 함수

### CASE 구문의 특징

#### ✓ CASE 구문의 특징

- IF~THEN~ELSE의 논리구조와 유사한 방식
- CASE 구문의 예와 DECODE

ELSE 생략 시 NULL 출력

```
CASE WHEN GRADE = 'A' THEN 'HIGH'
      WHEN GREDE ='B' TEHN 'MIDDLE '
      ELSE 'LOW'
END
```

```
CASE GRADE
  WHEN GRADE ='A' THEN 'HIGH'
  WHEN GRADE ='B' THEN 'MIDDLE'
  ELSE 'LOW'
END
```

```
DECODE (GRADE, 'A', 'HIGH', 'B', 'MIDDLE', 'LOW')
```

# 04. SQL 기본

## WHERE 절

### WHERE 절의 특징

#### ✓ WHERE 절의 목적

- SELECT, UPDATE, DELETE 문에서 조건을 설정하는 구문
- WHERE 절의 기본 형태
  - `SELECT * FROM user_t WHERE NAME = 'chris';`
  - `SELECT * FROM user_t WHERE NAME <> 'tommy';`
  - `UPDATE user_t SET AGE = 30 WHERE NAME = 'chris';`
  - `DELETE user_t WHERE name = 'tommy';`

# 04. SQL 기본

## WHERE 절

### WHERE 절의 연산자

#### ✓ WHERE 절에서 사용할 수 있는 다양한 연산자 (1)

- 비교 연산자 : = , < , <= , > , >=
- 부정의미의 비교 연산자 : != , ^= , <> , not 컬럼명 = 값 , not 컬럼명 > 값
- SQL 고유 연산자

- NOT BETWEEN A AND B  
- NOT IN (리스트)  
- IS NOT NULL

- BETWEEN A AND B : A 와 B 를 포함한 사이의 값 , IN (리스트) : 리스트 내의 값
- LIKE '비교 문자열' : 비교 문자열을 포함 (% : 문자열, \_ : 하나의 문자)
- IS NULL : NULL 값 확인 (※ NULL은 일반 비교연산자로 확인할 수 없음)

```
SELECT NAME, GRADE FROM user_t WHERE NAME LIKE 'chris%kim%';
```

```
SELECT NAME, JUMSU FROM user_t WHERE JUMSU in (100, 80, 70);
```



# 04. SQL 기본

## WHERE 절

### WHERE 절의 연산자

#### ✓ WHERE 절에서 사용할 수 있는 다양한 연산자 (2)

- 논리 연산자 : NOT, AND, OR

※ 연산자 우선순위 : 괄호 > 부정 연산자 > 비교 연산자 > 논리연산자

① 괄호

② NOT

③ 비교 연산자, SQL 연산자

④ AND

⑤ OR

[참고]

- 문자열 비교 : 첫 서로 다른 문자의 값 비교
- CHAR와 VARCHAR의 비교 : 길이가 다르면 길이가 긴 값이 크다고 판단

# 04. SQL 기본

## GROUP BY, HAVING 절

### GROUP BY 절의 특징과 집계 함수

#### ✓ GROUP BY 절의 특징

- 데이터를 그룹 별로 묶어서 처리하기 위해 사용
- 형식 : GROUP BY 기준 컬럼명 (기준 컬럼은 2개 이상이 될 수 있음)
- 집계 함수를 이용한 집계 데이터 도출 가능, ALIAS(별칭) 사용할 수 없음

#### ✓ 집계함수의 특징

- 집계함수의 개념 : 여러 행들의 그룹을 대상으로 하나의 결과를 반환하는 다중행 함수.
- 집계함수의 특징 :
  - GROUP BY 절이 꼭 있어야 함, WHERE 절에 사용할 수 없음
  - SELECT 절, HAVING절, ORDER BY 절에 사용할 수 있음

# 04. SQL 기본

## GROUP BY, HAVING 절

### GROUP BY 절의 특징과 집계 함수

#### ✓ 집계함수의 종류

- COUNT(\*) : NULL 값을 포함한 행의 개수
- COUNT(컬럼) : 값이 NULL인 행을 제외한 개수
- COUNT(DISTINCT 컬럼) : 값이 NULL이 아닌 행을 대상으로 중복을 제거한 개수
- SUM(컬럼) : 컬럼 값들의 합계
- AVG(컬럼) : 컬럼 값들의 평균
- MIN(컬럼) : 컬럼 값 중 최소값
- MAX(컬럼) : 컬럼 값 중 최대값

```
SELECT AVG(AGE) AS AVG_AGE FROM user_t;
```

# 04. SQL 기본

## GROUP BY, HAVING 절

### Having 절의 특징

#### ✓ Having 절의 특징

- 그룹을 나타내는 결과 집합의 행을 대상으로 조건 지정
- 일반적으로 GROUP BY 절 뒤에 위치하며, GROUP BY 절에서 사용된 컬럼을 조건으로 사용할 수 있음.

※ WHERE 절은 SELECT 문을 대상으로 조건 지정

```
SELECT JOB, COUNT(*) CNT, SUM(SAL) SAL  
FROM emp_t  
WHERE DEPT_NO IN ('1', '2', '3')  
GROUP BY JOB  
HAVING COUNT(*) > 2 AND SUM(SAL) > 5000
```

# 04. SQL 기본

## ORDER BY 절

### ORDER BY 절의 특징

#### ✓ ORDER BY 절의 특징

- SELECT 한 결과를 특정 컬럼을 기준으로 정렬하기 위해 사용하는 구문 (※ 기본 : 임의의 순으로 출력)
- ORDER BY 절에 컬럼을 지정하는 방식
  - 컬럼명, ALIAS(별칭), 컬럼 순서를 나타내는 정수
  - GROUP BY 절이 있을 경우 GROUP BY 대상 컬럼명만 지정 가능
  - 2 개 이상의 컬럼도 지정 가능
- ORDER BY 절의 정렬 옵션 : ASC (오름차순, 기본), DESC (내림차순)

<참고>

- 오라클은 NULL을 최대값으로 판단함
- NULL FIRST , NULL LAST 옵션으로 변경 가능

```
SELEECT NAME, POSITION FROM player_t WHERE NUMBER IS NOT NULL ORDER BY NAME DESC;
```



# 04. SQL 기본

## ORDER BY 절

### 〈참고〉 SELECT 문의 실행 순서

#### ✓ SELECT 관련 구문의 실행 순서

- 테이블에서 대상이 아닌 항목 제거 → 그룹핑 → 그룹핑 조건 적용 → 데이터 계산 및 출력 → 정렬

SELECT 칼럼명 AS “별칭”

FROM 테이블명

WHERE 조건식

GROUP BY 칼럼 (표현식)

HAVING 조건식

ORDER BY 칼럼 (표현식)

⑤ 계산 및 출력하고

① 테이블에서

② 대상이 아닌 것을 제외하고

③ 그룹핑 한 후

④ 그룹핑된 값이 조건에 맞는 데이터를

⑥ 정렬함

# 04. SQL 기본

## 조인

### 조인(Join)의 개념과 유형

#### ✓ 조인의 개념

- 둘 이상의 테이블을 연결하여 논리적 관계를 기준으로 검색하여 결과 집합을 만드는 기능
- 하나의 SQL 문으로 여러 테이블의 데이터를 연관하여 조회하는 기능

#### ✓ 조인의 유형 (1)

- EQUI Join : 테이블 간 컬럼 값들이 정확히 일치하는 경우에 사용, 대부분 PK와 FK 관계를 기반으로 함

➤ 형식 : SELECT 컬럼(들) FROM 테이블1 A, 테이블2 B WHERE A.컬럼명=B.컬럼명;

```
SELECT user_t.NAME USER_NAME, team_t.NAME TEAM_NAME FROM user_t, team_t  
WHERE user_t.TEAM_ID = team_t.TEAM_ID;
```

<참고>

- 동일한 컬럼명이 있을 경우 별칭 이용 가능

# 04. SQL 기본

## 조인

### 조인(Join)의 개념과 유형

- BETWEEN, >, >=, <, <= 등의 연산자

#### ✓ 조인의 유형 (2)

- Non EQUI JOIN : 테이블 간 컬럼 값들이 정확하게 일치하지 않은 경우에 사용, '=' 연산자가 아닌 다른 연산자를 사용.

```
SELECT E.ENAME EMP_NAME, E.SAL EMP_SAL, S.GRADE SAL_GRDE  
FROM EMPLOY E, SALGRADE S  
WHERE E.SAL BETWEEN S.LOWSAL AND S.HIGHSAL;
```

#### <참고>

- 3개 이상의 TABLE을 JOIN할 수도 있다. (WHERE 절에 2개 이상의 JOIN 조건이 필요하다.)
- EQUI 조인과 Non EQUI Join을 하나의 쿼리에서 함께 사용할 수 있다.

# 04. SQL 기본

## 조인

### 조인(Join)의 개념과 유형

#### ✓ 조인의 유형 (3)

- Inner Join : 행에 동일한 값이 있는 컬럼 조인 (기본 옵션), 조건에 충족되는 데이터만 출력

➤ 형식 : SELECT 컬럼(들) FROM 테이블1 A, 테이블2 B WHERE A.컬럼=B.컬럼;

SELECT 컬럼(들) FROM 테이블1 A INNER JOIN 테이블2 B ON A.컬럼=B.컬럼;

➤ 특징 : USING 혹은 ON 조건절 필수, CROSS JOIN 혹은 OUTER JOIN과 동시 사용할 수 없음,  
동일한 컬럼명이 있을 경우 ALIAS(별칭) 사용

- Natural Join : 두개의 테이블에서 같은 이름을 가진 컬럼들이 모두 동일한 데이터를 갖고 있는 경우에 수행되는 방식

SELECT \* FROM USER A NATURAL JOIN USER\_ACTION B;

## 04. SQL 기본

### 조인

#### 조인(Join)의 개념과 유형

##### ✓ 조인의 유형 (4)

- Outer Join : 상대 릴레이션에서 대응되는 튜플을 찾지 못하거나, NULL 값이 포함된 튜플들을 다루기 위한 조인
  - Left Outer Join : 왼쪽의 릴레이션을 기준으로 결과 릴레이션을 생성, 왼쪽 릴레이션의 튜플들을 모두 표시한 후 대응되는 값이 없을 경우 NULL로 대체.
  - Right Outer Join : 오른쪽 릴레이션을 기준으로 결과 릴레이션을 생성, 오른쪽 릴레이션의 튜플들을 모두 표시한 후 대응되는 값이 없을 경우 NULL로 대체.
  - Full Outer Join : 양쪽 릴레이션 모두를 기준으로 결과 릴레이션을 생성, 대응되는 값이 없을 경우 양쪽 모두 NULL로 대체.

## 04. SQL 기본

### 조인

#### 조인(Join)의 개념과 유형

##### ✓ 조인의 유형 (5)

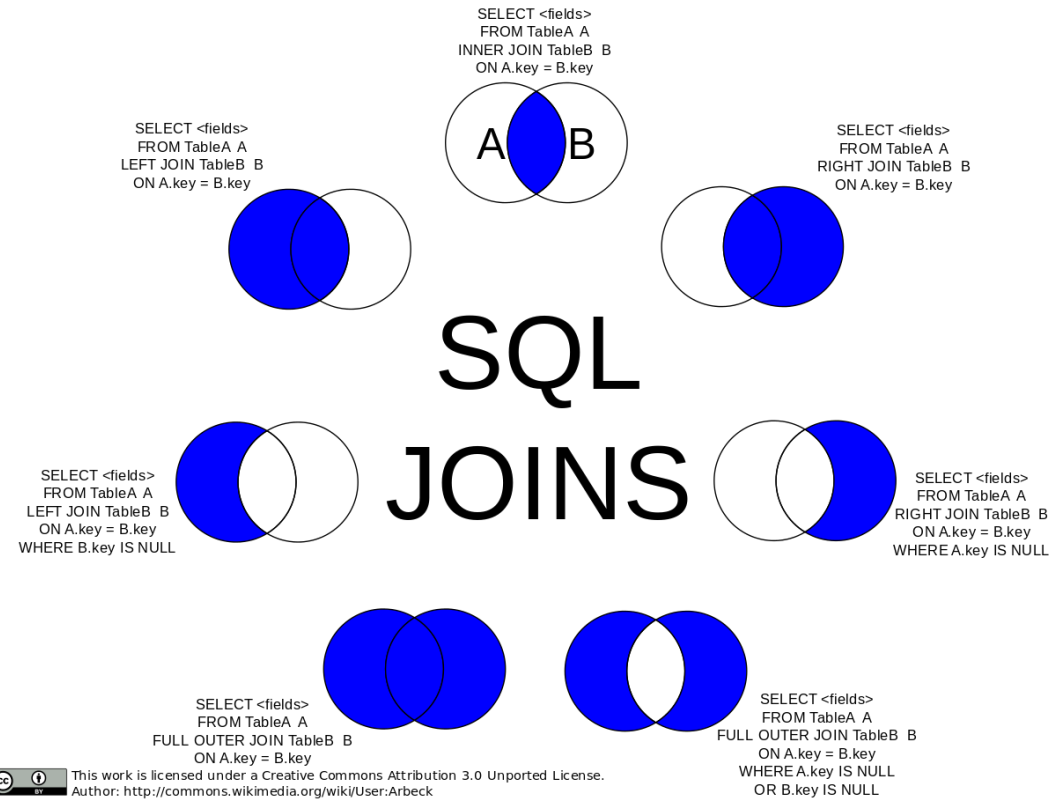
- Cross Join : CROSS JOIN, 연결된 테이블의 Cartesian Product를 반환 (모든 조합)
- Nested Loop Join : 선행 테이블을 하나씩 접근하며 추출한 값으로 연결할 테이블을 조인
- Sort Merge Join : 대상 테이블의 처리 범위를 각자 접근 및 정렬한 결과를 차례로 스캔하며 연결 조건으로 머지(Merge)

# 04. SQL 기본

## 조인

### 조인(Join)의 개념과 유형

#### ✓ 조인의 유형 정리



# SQLD

## 자격증 특강

컴퓨터·AI공학과  
김윤수 교수





## 05. SQL 활용

### SQL 활용

- ✓ 서브쿼리
- ✓ 집합 연산자
- ✓ 그룹 함수
- ✓ 윈도우 함수
- ✓ Top N 쿼리
- ✓ 계층형 질의와 셀프 조인
- ✓ PIVOT 절과 UNPIVOT 절
- ✓ 정규 표현식

# 05. SQL 활용

## 서브쿼리

### 서브쿼리(Subquery)의 개념과 종류

#### ✓ 서브쿼리의 개념

- 메인 쿼리에 포함된 종속 관계의 쿼리, 하나의 SQL문 안에 포함된 또 다른 SQL문

#### ✓ 서브쿼리의 종류

단일 행 서브쿼리 (=, <, >, <=, >=, <>) , 다중 행 서브쿼리 (IN, ANY 등), 다중 컬럼 서브쿼리

- Nested Subquery : WHERE 절에 위치. 1개 이상의 컬럼 또는 1개 이상의 행을 반환.
  - 비연관 서브쿼리 : 메인쿼리와 관계가 없음 (즉, 메인쿼리의 컬럼이 존재하지 않음)
  - 연관 서브쿼리 : 메인쿼리와 관계가 있음 (즉, 메인쿼리의 컬럼이 존재함)
- Inline View : FROM 절에 위치. 뷰(View)와 동일한 방식으로 사용 (임시 뷰)
- Scalar Subquery : SELECT 절에 위치. 단일 컬럼, 단일 행을 반환.

※ 반환 데이터 형태에 따라 '단일행 서브쿼리, 다중행 서브쿼리, 다중컬럼 서브쿼리' 로 구분 할 수 있음

## 05. SQL 활용

### 서브쿼리

#### 서브쿼리(Subquery)의 개념과 종류

##### ✓ 서브쿼리의 개념

- 메인 쿼리에 포함된 종속 관계의 쿼리, 하나의 SQL문 안에 포함된 또 다른 SQL문

##### ✓ 서브쿼리의 종류

- Nested Subquery : Having절, WHERE 절에 위치. 1개 이상의 칼럼 또는 1개 이상의 행을 반환.
- Inline View : FROM 절에 위치. 뷰(View)와 동일한 방식으로 사용 (임시 뷰)
- Scalar Subquery : SELECT 절에 위치. 단일 컬럼, 단일 행을 반환.

※ 반환 데이터 형태에 따라 '단일행 서브쿼리, 다중행 서브쿼리, 다중칼럼 서브쿼리' 로 구분 할 수 있음

<예>

```
SELECT name, position FROM t_player  
WHERE team_id = (SELECT team_id FROM t_player WHERE name = 'chris')  
ORDER BY name
```

# 05. SQL 활용

## 서브쿼리

### 서브쿼리(Subquery)의 개념과 종류

#### ✓ 서브쿼리의 예

```
SELECT a.empno
      , a.ename
      , b.mgr_name
      , (SELECT dd.dname
          FROM dept dd
          WHERE dd.deptno = a.deptno) AS dept_name
FROM emp a
      , (SELECT a.empno AS mgr_no
          , a.ename AS mgr_name
          , b.dname AS mgr_dept
          FROM emp a
          , dept b
          WHERE a.deptno = b.deptno) b
WHERE a.mgr = b.mgr_no
AND a.deptno IN (SELECT DISTINCT aa.deptno
                 FROM emp aa
                 WHERE aa.job = 'MANAGER')
```

스칼라 서브 쿼리

인라인 뷰

중첩 서브 쿼리

# 05. SQL 활용

## 〈추가〉 뷰(View)

### 뷰(View)의 개념과 특징

#### ✓ 뷰의 개념

- 하나 이상의 기본 테이블이나 다른 뷰를 이용하여 생성된 가상 테이블.
- 데이터베이스내에 뷰의 정의만 저장하고, 디스크 저장공간 할당은 안됨

장점 : 보안성, 편의성

#### ✓ 뷰의 구문

- CREATE (or REPLACE) VIEW 뷰이름 AS 서브쿼리

```
CREATE VIEW other_view  
AS SELECT name, age, deptno FROM user_t;
```

- DROP VIEW 뷰이름

## 05. SQL 활용

### 〈추가〉 뷰(View)

#### 뷰(View)의 종류

##### ✓ 뷰의 종류

| 뷰의 종류                  | 설명  |
|------------------------|---|
| 단순 뷰<br>(Simple View)  | <ul style="list-style-type: none"><li>- 하나의 기본 테이블에 의해 정의된 뷰로 기본 테이블과 동일한 DML 문 사용</li><li>- 뷰에 대한 무결성 제약조건도 기본 테이블의 제약조건이 적용됨</li></ul>  |
| 복합 뷰<br>(Complex View) | <ul style="list-style-type: none"><li>- 두개 이상의 기본 테이블로 구성된 뷰</li><li>- 무결성 제약조건, 표현식, GROUP BY절의 유무에 따라 DML 명령문을 제한적으로 사용</li><li>- DISTINCT, 그룹함수, GROUP BY, ROWNUM 등을 포함할 수 없음.</li></ul> |
| 인라인 뷰<br>(Inline View) | <ul style="list-style-type: none"><li>- FROM 절에서 참조하는 테이블의 크기가 클 경우, 필요한 행과 컬럼만으로 구성된 집합을 재정의하여 질의문을 효율적으로 구성</li><li>- FROM 절에서 서브쿼리를 사용하여 생성한 임시 뷰로 SQL 명령문이 실행되는 동안만 임시적으로 사용</li></ul>  |

# 05. SQL 활용

## 집합 연산자

### 집합 연산자의 개념과 특징

#### ✓ 집합 연산자의 개념

- 두 개 이상의 테이블에 대한 쿼리의 결과 집합을 대상으로 연산을 수행
- 조인을 사용하지 않고 연관된 데이터를 조회하는 방법 중 하나

#### ✓ 집합 연산자의 종류

- UNION : 쿼리 결과의 합집합, 중복된 행은 하나의 행으로 출력
- UNION ALL : 쿼리 결과의 합집합, 중복된 행도 그대로 출력 (질의 결과가 상호 배타적일 때 주로 사용)
- INTERSECT : 쿼리 결과의 교집합, 중복된 행은 하나의 행으로 출력
- MINUS / EXCEPT : 앞의 쿼리 결과에서 뒤의 쿼리 결과를 뺀 차집합, 중복된 행은 하나의 행으로 출력

```
SELECT * FROM user_t UNION ALL SELECT * from student_t;
```

```
SELECT * FROM user_t MINUS SELECT * from student_t;
```

## 05. SQL 활용

### 그룹 함수

#### 그룹 함수의 특징과 종류

##### ✓ 그룹 함수의 특징과 종류

- SQL에서 사용할 수 있는 데이터 분석 함수
- GROUP BY 후 적용할 수 있는 분석 함수
- 그룹 함수의 종류
  - 집계 함수 : COUNT, SUM, AVG, MAX, MIN 등
  - 총계 함수 : ROLLUP, CUBE, GROUPING SETS, GROUPING 등



# 05. SQL 활용

## 그룹 함수

### 그룹 함수의 특징과 종류

#### ✓ 그룹 함수의 종류별 특징 (1)

- ROLLUP : GROUP BY로 묶인 컬럼의 소계(Subtotal)를 계산, 계층 구조, GROUP BY 컬럼의 순서가 변경되면 결과도 변경.  
Grouping Column의 개수가 N이면 N+1 Level의 소계가 생성됨.

➤ 형식 : ROLLUP(A, B) : 2개의 소그룹을 지정한 경우로 각각 A,B로 그룹핑한 소계, A로 그룹핑한 소계, 총합계를 출력.

| Year | Month | Product   | Amount |
|------|-------|-----------|--------|
| 2023 | 1     | Product A | 100    |
| 2023 | 1     | Product B | 200    |
| 2023 | 2     | Product A | 150    |
| 2023 | 2     | Product B | 300    |

```
SELECT Year, Month, Product, SUM(Amount) AS TotalSales
FROM Sales
GROUP BY ROLLUP(Year, Month, Product);
```



| Year | Month | Product   | TotalSales |
|------|-------|-----------|------------|
| 2023 | 1     | Product A | 100        |
| 2023 | 1     | Product B | 200        |
| 2023 | 1     | NULL      | 300        |
| 2023 | 2     | Product A | 150        |
| 2023 | 2     | Product B | 300        |
| 2023 | 2     | NULL      | 450        |
| 2023 | NULL  | NULL      | 750        |
| NULL | NULL  | NULL      | 750        |

# 05. SQL 활용

## 그룹 함수

### 그룹 함수의 특징과 종류

#### ✓ 그룹 함수의 종류별 특징 (2)

- CUBE : 결합 가능한 모든 조합에 대한 다차원 집계를 생성
  - 형식 : CUBE (A, B) : 2개의 소그룹을 지정한 경우로 각각 A,B로 그룹핑한 소계, A로 그룹핑한 소계, B로 그룹핑한 소계, 총합계를 출력.

| Year | Month | Product   | Amount |
|------|-------|-----------|--------|
| 2023 | 1     | Product A | 100    |
| 2023 | 1     | Product B | 200    |
| 2023 | 2     | Product A | 150    |
| 2023 | 2     | Product B | 300    |



| Year | Month | Product   | TotalSales |
|------|-------|-----------|------------|
| 2023 | 1     | Product A | 100        |
| 2023 | 1     | Product B | 200        |
| 2023 | 1     | NULL      | 300        |
| 2023 | 2     | Product A | 150        |
| 2023 | 2     | Product B | 300        |
| 2023 | 2     | NULL      | 450        |
| 2023 | NULL  | NULL      | 750        |
| NULL | 1     | Product A | 100        |
| NULL | 1     | Product B | 200        |
| NULL | 1     | NULL      | 300        |
| NULL | 2     | Product A | 150        |
| NULL | 2     | Product B | 300        |
| NULL | 2     | NULL      | 450        |
| NULL | NULL  | Product A | 250        |
| NULL | NULL  | Product B | 500        |
| NULL | NULL  | NULL      | 750        |

```
SELECT Year, Month, Product, SUM(Amount) AS TotalSales
FROM Sales
GROUP BY CUBE(Year, Month, Product);
```

# 05. SQL 활용

## 그룹 함수

### 그룹 함수의 특징과 종류

#### ✓ 그룹 함수의 종류별 특징 (3)

- GROUPING SETS : 특정 항목 (원하는 부분)의 소계를 계산, 인수의 순서가 바뀌어도 동일한 결과

➤ 형식 : GROUPING SETS (A, B, { }) : A로 그룹핑한 소계, B로 그룹핑한 소계, 총 합계

GROUPING SETS (A, ROLLUP(B, C)) : ROLLUP과 CUBE도 인자로 사용할 수 있음

| Year | Month | Product   | Amount |
|------|-------|-----------|--------|
| 2023 | 1     | Product A | 100    |
| 2023 | 1     | Product B | 200    |
| 2023 | 2     | Product A | 150    |
| 2023 | 2     | Product B | 300    |



| Year | Month | Product   | TotalSales |
|------|-------|-----------|------------|
| 2023 | 1     | NULL      | 300        |
| 2023 | 2     | NULL      | 450        |
| NULL | 1     | Product A | 100        |
| NULL | 1     | Product B | 200        |
| NULL | 2     | Product A | 150        |
| NULL | 2     | Product B | 300        |
| 2023 | NULL  | NULL      | 750        |

```
SELECT Year, Month, Product, SUM(Amount) AS TotalSales
FROM Sales
GROUP BY GROUPING SETS ((Year, Month), (Month, Product), (Year));
```

# 05. SQL 활용

## 그룹 함수

### 그룹 함수의 특징과 종류

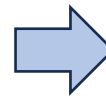
#### ✓ 그룹 함수의 종류별 특징 (4)

- GROUPING : 그룹 함수에서 생성된 합계를 구분해주는 함수, 소계 여부에 따라 1, 0 혹은 원하는 문자열을 표시할 수 있음.

➤ 참고 : ROLLUP, CUBE, GROUPING SETS과 함께 쓰임, 결과값에 따라 CASE 혹은 DECODE 표시할 문자열 선택

| Year | Month | Product   | Amount |
|------|-------|-----------|--------|
| 2023 | 1     | Product A | 100    |
| 2023 | 1     | Product B | 200    |
| 2023 | 2     | Product A | 150    |
| 2023 | 2     | Product B | 300    |

```
SELECT
  DECODE(GROUPING(Year), 1, 'All Years', TO_CHAR(Year)) AS Year,
  DECODE(GROUPING(Month), 1, 'All Months', TO_CHAR(Month)) AS
  Month, SUM(Amount) AS TotalSales
FROM Sales
GROUP BY ROLLUP (Year, Month);
```



| Year      | Month      | TotalSales |
|-----------|------------|------------|
| 2023      | 1          | 300        |
| 2023      | 2          | 450        |
| 2023      | All Months | 750        |
| All Years | All Months | 750        |

## 05. SQL 활용

### 윈도우 함수

#### 윈도우 함수의 특징과 종류

##### ✓ 윈도우 함수의 특징과 종류

- OVER 문구를 키워드로 포함하며, 행과 행 간의 관계를 정의하기 위해 사용
- 윈도우 함수의 종류
  - 순위 함수 : RANK, DENSE\_RANK, ROW\_NUMBER
  - 집계 함수 : SUM, MAX, MIN, AVG, COUNT
  - 행 순서 함수 : FIRST\_VALUE, LAST\_VALUE, LAG, LEAD
  - 비율 함수 : CUME\_DIST, PERCENT\_RANK, NTILE, RATIO\_TO\_REPORT

# 05. SQL 활용

## 윈도우 함수

### 윈도우 함수의 종류와 특징 - 순위 함수

#### ✓ 윈도우 함수의 종류와 특징 - 순위 함수

- RANK : 순위를 카운트, 중복 순위를 포함 (즉, 중복된 수 만큼 증가한 후 그 다음 카운트 시작)
- DENSE\_RANK : 순위를 카운트, 중복 순위를 무시
- ROW\_NUMBER : 동일한 값(순위)라도 고유한 순위를 부여

```
SELECT Salesperson, SUM(Amount) AS TotalSales,  
       RANK() OVER (ORDER BY SUM(Amount) DESC) AS SalesRank,  
       DENSE_RANK() OVER (ORDER BY SUM(Amount) DESC) AS SalesDenseRank,  
       ROW_NUMBER() OVER (ORDER BY SUM(Amount) DESC) AS SalesRowNumber  
FROM Sales  
GROUP BY Salesperson;
```



| Salesperson | TotalSales | SalesRank | SalesDenseRank | SalesRowNumber |
|-------------|------------|-----------|----------------|----------------|
| John        | 1000       | 1         | 1              | 1              |
| Jane        | 800        | 2         | 2              | 2              |
| Bob         | 800        | 2         | 2              | 3              |
| Alice       | 500        | 4         | 3              | 4              |

## 05. SQL 활용

### 윈도우 함수

#### 윈도우 함수의 종류와 특징 - 집계 함수

##### ✓ 윈도우 함수의 종류와 특징 - 집계 함수

- SUM (합계), MAX(최대값), MIN (최소값), AVG(평균값), COUNT(건수)
- 데이터를 그룹화 하지 않고 각 행에 대해 연산을 수행할 수 있음, 윈도우 구문을 이용하여 대상 행의 범위를 지정할 수 있음

```
SELECT Salesperson, Month, Amount,  
       SUM(Amount) OVER (PARTITION BY Salesperson) AS TotalSales,  
       MAX(Amount) OVER (PARTITION BY Salesperson) AS MaxSale,  
       AVG(Amount) OVER (PARTITION BY Salesperson) AS AvgSale,  
       COUNT(*) OVER (PARTITION BY Salesperson) AS SaleCount  
FROM Sales;
```



| Salesperson | Month | Amount | TotalSales | MaxSale | AvgSale | SaleCount |
|-------------|-------|--------|------------|---------|---------|-----------|
| John        | 1     | 200    | 600        | 300     | 200     | 3         |
| John        | 2     | 300    | 600        | 300     | 200     | 3         |
| John        | 3     | 100    | 600        | 300     | 200     | 3         |
| Jane        | 1     | 150    | 450        | 200     | 150     | 3         |
| Jane        | 2     | 100    | 450        | 200     | 150     | 3         |
| Jane        | 3     | 200    | 450        | 200     | 150     | 3         |

## 05. SQL 활용

### 윈도우 함수

#### 윈도우 함수의 종류와 특징 - 집계 함수

#### ✓ 윈도우 함수의 종류와 특징 - 집계 함수 (계속)

```
SELECT  
  Salesperson,  
  Date,  
  Amount,  
  SUM(Amount) OVER (  
    PARTITION BY Salesperson  
    ORDER BY Date  
    RANGE BETWEEN INTERVAL '1' MONTH PRECEDING AND CURRENT  
  ROW  
  ) AS RollingTotalSales  
FROM Sales;
```



| Salesperson | Date       | Amount | RollingTotalSales |
|-------------|------------|--------|-------------------|
| John        | 2024-01-10 | 200    | 200               |
| John        | 2024-01-20 | 300    | 500               |
| John        | 2024-02-05 | 100    | 400               |
| Jane        | 2024-01-11 | 150    | 150               |
| Jane        | 2024-01-23 | 100    | 250               |
| Jane        | 2024-02-06 | 200    | 300               |



## 05. SQL 활용

### 윈도우 함수

#### 윈도우 함수의 종류와 특징 - 행 순서 함수

##### ✓ 윈도우 함수의 종류와 특징 - 행 순서 함수

- FIRST\_VALUE (가장 앞의 데이터), LAST\_VALUE (가장 마지막의 데이터)
- LAG (특정 수 만큼 앞의 데이터), LEAD (특정 수 만큼 뒤의 데이터)

```
SELECT Salesperson, Date, Amount,  
       FIRST_VALUE(Amount) OVER (  
         PARTITION BY Salesperson ORDER BY Date  
       ) AS FirstSaleAmount,  
       LEAD(Amount, 1) OVER (  
         PARTITION BY Salesperson ORDER BY Date  
       ) AS NextSaleAmount  
FROM Sales;
```



| Salesperson | Date       | Amount | FirstSaleAmount | NextSaleAmount |
|-------------|------------|--------|-----------------|----------------|
| John        | 2024-01-10 | 200    | 200             | 300            |
| John        | 2024-01-20 | 300    | 200             | 100            |
| John        | 2024-02-05 | 100    | 200             | NULL           |
| Jane        | 2024-01-11 | 150    | 150             | 100            |
| Jane        | 2024-01-23 | 100    | 150             | 200            |
| Jane        | 2024-02-06 | 200    | 150             | NULL           |

## 05. SQL 활용

### 윈도우 함수

#### 윈도우 함수의 종류와 특징 - 비율 함수

- ✓ 윈도우 함수의 종류와 특징 - 비율 함수
- **RATIO\_TO\_PERCENT** : 파티션 합계에서 차지하는 비율 계산
  - **PERCENT\_RANK** : 현재 행의 백분위 수를 계산 (0 ~ 1을 구간으로 함)
  - **CUME\_DIST** : 누적 백분율을 계산 (0 ~ 1 사이의 값)
  - **NTILE** : 행들을 N 등분 후 현재 행의 등급을 계산

```
SELECT Salesperson, Amount,  
       RATIO_TO_REPORT(Amount) OVER () AS RatioToReport,  
       PERCENT_RANK() OVER (ORDER BY Amount) AS PercentRank  
FROM Sales;
```



| Salesperson | Amount | RatioToReport | PercentRank |
|-------------|--------|---------------|-------------|
| John        | 100    | 0.1           | 0.0         |
| Jane        | 200    | 0.2           | 0.333       |
| Bob         | 300    | 0.3           | 0.666       |
| Alice       | 400    | 0.4           | 1.0         |

# 05. SQL 활용

## Top N 쿼리

### TOP N 쿼리의 특징과 방식

#### ✓ TOP N 쿼리의 특징과 방식

- 테이블에서 상위 N 개의 행을 검색하는 쿼리
- TOP N 쿼리 실행 방식
  - ROWNUM : 오라클에서 사용하는 의사 컬럼 ( '=' 조건은 사용할 수 없으며, < 혹은 <= 조건을 이용한다.)
  - FETCH FIRST N ROWS ONLY : ANSI SQL에서 사용하는 구문 (ORDER BY 절과 함께 사용)

```
SELECT column_name  
FROM (  
    SELECT column_name  
    FROM table_name  
    ORDER BY column_name DESC  
)  
WHERE ROWNUM <= N;
```

```
SELECT column_name  
FROM table_name  
ORDER BY column_name DESC  
FETCH FIRST N ROWS ONLY;
```

## 05. SQL 활용

### 계층형 질의와 셀프 조인

#### 계층형 질의와 셀프 조인의 특징

##### ✓ 계층형 질의

- 계층형 데이터를 조회하기 위해 사용
  - 엔터티를 순환관계 데이터 모델로 설계할 경우 발생 (예: 조직, 메뉴, 부품 등)
  - START WITH ... CONNECT BY 구문을 사용

##### ✓ 셀프 조인 (Self Join)

- 동일한 테이블 간 조인
  - 한 테이블 내 컬럼간 연관 관계가 있을 경우에 사용, ALIAS(별칭) 사용 필수

# 05. SQL 활용

## 계층형 질의와 셀프 조인

### 계층형 질의와 셀프 조인의 특징

#### ✓ 계층형 질의와 셀프 조인의 예

```
SELECT employee_id, first_name, manager_id  
FROM employees  
START WITH manager_id IS NULL  
CONNECT BY PRIOR employee_id = manager_id  
ORDER SIBLINGS BY first_name;
```

manager\_id가 NULL인 최상위 직원(보통 최고경영자)부터 시작해서, 각 직원의 employee\_id가 다른 직원의 manager\_id와 일치하는 모든 직원을 재귀적으로 선택

employees 테이블을 두 번 참조하여 (e로 한 번, m으로 한 번), 각 직원의 이름과 그들의 직속 상사의 이름을 조회. LEFT JOIN을 사용하므로 상사가 없는 직원(예: 최고경영자)도 결과에 포함

```
SELECT e.employee_id, e.first_name AS Employee, m.first_name AS  
Manager  
FROM employees e  
LEFT JOIN employees m ON e.manager_id = m.employee_id;
```



| employee_id | Employee  | Manager |
|-------------|-----------|---------|
| 100         | Steven    | NULL    |
| 101         | Neena     | Steven  |
| 102         | Lex       | Steven  |
| 103         | Alexander | Neena   |
| ...         | ...       | ...     |

## 05. SQL 활용

### PIVOT 절과 UNPIVOT 절, 정규 표현식

#### PIVOT과 UNPIVOT, 정규 표현식의 특징

- ✓ PIVOIT과 UNPIVOT의 특징
  - PIVOT : 행을 열로 변환하여 데이터를 요약하기 위해 사용.
  - UNPIVOT : 열을 행으로 변환하여 데이터를 요약하기 위해 사용.
- ✓ 정규 표현식의 특징
  - 문자열의 특정 패턴을 기술하는 방법

```
SELECT * FROM table_name  
WHERE REGEXP_LIKE(column_name, '^[A-Za-z0-9]+@[A-Za-z0-9]+\.[A-Za-z]{2,}$');
```

# SQLD

## 자격증 특강

컴퓨터·AI공학과  
김윤수 교수



## 06. 기출문제 동향과 풀이

### 기출문제 동향과 풀이

- ✓ 기출문제 동향
- ✓ 주제별 문제 풀이





## 06. 기출문제 동향과 풀이

### 기출문제 동향

- ✓ 데이터자격검정 : <https://www.dataq.or.kr/>
- ✓ 선택형 50문항, 문항당 2점

| 과목명         | 과목별 세부 항목                  | 문항수 | 배점  | 검정시간 | 합격기준                      |
|-------------|----------------------------|-----|-----|------|---------------------------|
| 데이터 모델링의 이해 | 데이터 모델링의 이해<br>데이터 모델과 SQL | 10  | 20점 | 90분  | 60점 이상<br>(과목별 40% 미만 과락) |
| SQL 기본 및 활용 | SQL 기본<br>SQL 활용<br>관리 구문  | 40  | 80점 |      |                           |

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

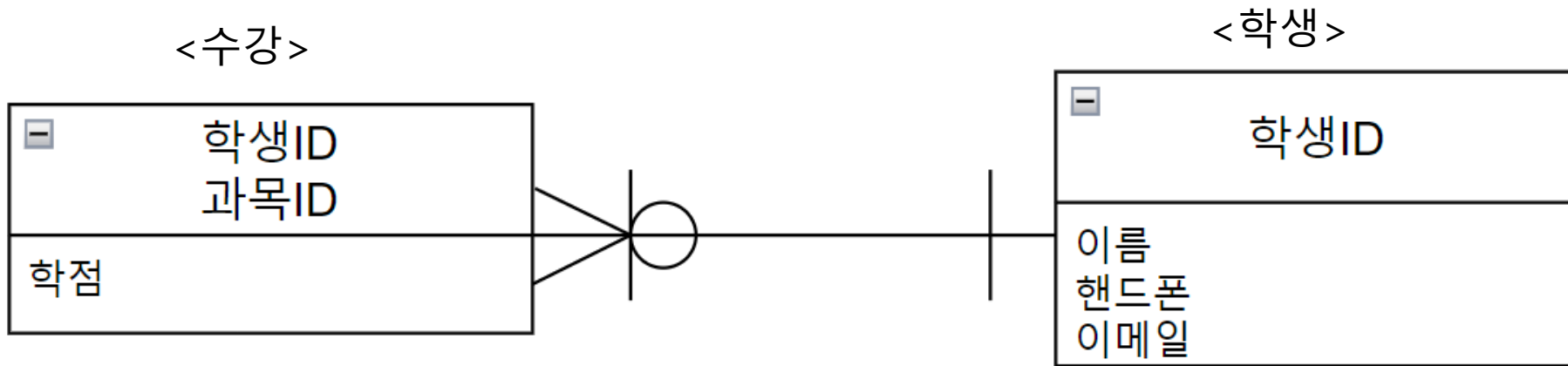
문) 데이터 모델링 과정 중 논리적 데이터 모델링에 대한 설명으로 가장 적절한 것은 무엇인가?

- ① 데이터 이상 현상 제거를 위한 정규화를 수행
- ② 업무 중심으로 포괄적인 모델링을 수행
- ③ 실제 DBMS의 특성을 반영
- ④ 추상화 수준이 가장 높은 모델링

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 ERD 관계에 대한 설명으로 잘못된 것은 무엇인가?



- ① 수강내역은 하나의 학생을 갖는다.
- ② 한 명의 학생은 반드시 수강내역을 갖는다.
- ③ 한 명의 학생은 여러 개의 수강 내역을 가질 수 있다.
- ④ 테이블로 변환 시 수강 내역 테이블은 학생 테이블의 PK(주식별자)인 학생 ID를 FK(외래 식별자)로 가진다.

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 데이터베이스 3층 스키마에 대한 설명으로 적절하지 않은 것은 무엇인가?

- ① 외부 스키마, 개념 스키마, 내부 스키마로 구성된다.
- ② 외부 단계는 응용 프로그래머가 접근하는 DB 관점을 정의한다.
- ③ 내부 단계는 DB가 물리적으로 저장된 형식을 표현한다.
- ④ 개념 단계는 사용자 관점에서 접근한다.



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음은 사원 엔터티의 주소에 대한 설명이다. 어떠한 속성에 속하는가?

-----

주소는 도, 시, 구, 도로명 으로 구성된다.

-----

- ① 단일치 속성
- ② 다중치 속성
- ③ 복합 속성
- ④ 설계 속성

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 회사 업무 관련 테이블 중에서 자주 참조하는 사원 컬럼과 부서 컬럼을 추출하여 데이터 입/출력 속도를 줄일 수 있었다.

이와 관련된 반정규화 기법으로 가장 적절한 것은 무엇인가?

- ① 수직 분할
- ② 수직 분할
- ③ 테이블 분리
- ④ 중복 데이터 추가

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 보기에서 설명하는 정규화로 가장 적절한 것은 무엇인가?

-----

기본 키를 제외한 일반 속성 간에 종속성이 있을 경우 별도의 테이블로 분할한다.

-----

- ① 제 1정규화
- ② 제 2정규화
- ③ 제 3정규화
- ④ 반 정규화



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음에서 설명하는 용어로 가장 적절한 것은 무엇인가?

-----

학점 속성의 값으로는 A, B, C, D, F의 5가지가 있다. 이와 같이 특정한 속성 값이 가질 수 있는 값의 범위를 의미한다.

-----

① 시스템카탈로그

② 다중값 속성

③ 차수

④ 도메인



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 식별자는 그 특징에 따라 여러가지로 구분할 수 있다. 다음 내용에 해당하는 것으로 가장 적절한 것은 무엇인가?

-----

유일성은 만족하나 최소성을 만족하지 않는 키를 의미한다.

-----

- ① 기본키
- ② 슈퍼키
- ③ 외래키
- ④ 후보키

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 지문이 설명하는 용어로 가장 적절한 것은 무엇인가?

-----

업무적으로 만들어지지는 않았으나 원조 식별자가 복잡한 구성을 갖기 때문에 인위적으로 만들어 사용하는 식별자를 의미한다.

-----

- ① 외부식별자
- ② 내부식별자
- ③ 본질식별자
- ④ 인조식별자

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음은 데이터 모델링의 관점 중 어떤 관점을 설명하는 것인가?

-----

업무의 실제 진행 하는 방식, 진행하는 일에 대하여 모델링한다.

-----

- ① 프로세스 관점
- ② 데이터 관점
- ③ 데이터와 프로세스 관점
- ④ 업무 중심 관점



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음과 같은 조건에서 실행된 SQL의 결과로 가장 적절한 것은 무엇인가?

```
-----  
SELECT employee_id, employee_name, manager_id  
FROM employees  
START WITH manager_id = 1  
CONNECT BY PRIOR employee_id = manager_id;  
-----
```

※ 보기 구문 생략

테이블 데이터 예시

| employee_id | employee_name | manager_id |
|-------------|---------------|------------|
| 1           | John Doe      | NULL       |
| 2           | Jane Smith    | 1          |
| 3           | Mike Brown    | 1          |
| 4           | Karen White   | 2          |

예상 쿼리 실행 결과

| employee_id | employee_name | manager_id |
|-------------|---------------|------------|
| 2           | Jane Smith    | 1          |
| 3           | Mike Brown    | 1          |
| 4           | Karen White   | 2          |

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 SQL 문의 실행 결과에 대하여 가장 적절하게 설명한 것은 무엇인가?

-----  
SELECT USER\_NAME, NVL2(GRADE, '학점취득', '학점미취득') AS GRADE\_CHECK FROM user\_t;  
-----

- ① GRADE에 값이 없을 경우 오류가 발생한다.
- ② GRADE에 값이 있을 경우 GRADE의 값을 표시한다.
- ③ GRADE가 1 일 경우 '학점취득'을 표시한다.
- ④ GRADE에 값이 있을 경우 '학점취득'을 표시한다.

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 기능을 수행하는 DDL 문으로 가장 적절한 것은 무엇인가?

-----

user\_t 테이블의 TEL 컬럼의 조건을 NOT NULL로 변경한다.

-----

- ① ALTER TABLE user\_t SET TEL NOT NULL;
- ② ALTER TABLE user\_t MODIFY TEL NOT NULL;
- ③ ALTER TABLE user\_t UPDATE TEL NOT NULL;
- ④ ALTER TABLE user\_t CONSTRAINT TEL NOT NULL;



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음과 같은 조건에서 실행된 SQL의 결과로 가장 적절한 것은 무엇인가?

-----

```
SELECT employee_id, employee_name, department_code, managed_department_code
```

```
FROM employees
```

```
WHERE department_code = managed_department_code;
```

-----

※ 보기 구문 생략

테이블 데이터 예시

| employee_id | employee_name | department_code | managed_department |
|-------------|---------------|-----------------|--------------------|
| 1           | John Doe      | D001            | D002               |
| 2           | Jane Smith    | D002            | D002               |
| 3           | Mike Brown    | D001            | NULL               |
| 4           | Karen White   | D003            | D003               |

예상 쿼리 실행 결과

| employee_id | employee_name | department_code | managed_department |
|-------------|---------------|-----------------|--------------------|
| 2           | Jane Smith    | D002            | D002               |
| 4           | Karen White   | D003            | D003               |



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 중 그 종류가 다른 SQL 문은 어떤 것인가?

- ① CREATE
- ② ALTER
- ③ DROP
- ④ SELECT





## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 SQL과 동일한 결과를 갖는 SQL은 무엇인가?

-----

```
SELECT Year, Month, Product, SUM(Amount) AS TotalSales
```

```
FROM Sales
```

```
GROUP BY CUBE(Year, Product);
```

-----

※ 보기 구문 생략

```
SELECT Year, Month, Product, SUM(Amount) AS TotalSales
FROM Sales
GROUP BY GROUPING SETS (
    (Year, Product),
    (Year),
    (Product),
    ()
);
```



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음과 같은 조건에서 실행된 SQL의 결과로 가장 적절한 것은 무엇인가?

-----

```
SELECT employee_name FROM employees
```

```
UNION ALL
```

```
SELECT department_name FROM departments
```

```
MINUS
```

```
SELECT department_name FROM departments WHERE department_name = 'HR';
```

-----

※ 보기 구문 생략

Employees Table

| employee_id | employee_name | department_id |
|-------------|---------------|---------------|
| 1           | John Doe      | 101           |
| 2           | Jane Smith    | 102           |
| 3           | Mike Brown    | 103           |

Departments Table

| department_id | department_name |
|---------------|-----------------|
| 101           | HR              |
| 102           | IT              |
| 103           | Sales           |

결과 세트

| name       |
|------------|
| John Doe   |
| Jane Smith |
| Mike Brown |
| IT         |
| Sales      |



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 윈도우 함수 중 그 종류가 다른 것은 무엇인가?

- ① RANK
- ② DENSE\_RANK
- ③ ROW\_NUMBER
- ④ CUME\_DIST

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 윈도우 함수에 대한 설명 중 잘못 된 것은 무엇인가?

- ① PERCENT\_RANK : 백분율 순서
- ② CUME\_DIST : 현재 행이 차지하는 순위
- ③ RATIO\_TO\_REPORT : 총합계에 대한 값의 백분율
- ④ ROW\_NUMBER : 단순히 행 번호 표시, 값에 무관하게 고유한 순위 부여

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 학년별 평균키가 170 이상인 학년의 이름과 해당 학년의 평균치, 최대키를 출력하는 SQL의 예이다.

괄호 ( ) 안에 가장 적절한 구문은 무엇인가?

-----

```
SELECT team_name, AVG(height) AS average_height, MAX(height) AS max_height
```

```
FROM teams GROUP BY team_name, grade ( );
```

-----

① WHERE AVG(height) >= 170;

② HAVING AVG(height) >= 170;

③ ON AVG(height) >= 170;

④ WITH AVG(height) >= 170;

## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) ROLLBACK 에 대한 설명 중 가장 적절하지 않은 것은 무엇인가?

- ① INSERT 문 수행 후 ROLLBACK 시 입력이 취소된다.
- ② DELETE 문 수행 후 ROLLBACK 시 삭제가 취소된다.
- ③ UPDATE 문 수행 후 ROLLBACK 시 변경이 취소된다.
- ④ CREATE TABLE 문 수행 후 ROLLBACK 시 TABLE 생성이 취소된다.



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 SQL 문을 실행할 경우 출력되는 내용은 몇 건인가?

-----

```
SELECT e.employee_id, e.employee_name, d.department_id, d.department_name  
FROM employees e CROSS JOIN departments d;
```

-----

① 2

② 4

③ 6

④ 8

Employees Table

| employee_id | employee_name |
|-------------|---------------|
| 1           | John Doe      |
| 2           | Jane Smith    |

Departments Table

| department_id | department_name |
|---------------|-----------------|
| A             | HR              |
| B             | IT              |



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 SQL 문의 실행 결과로 가장 적절한 것은 무엇인가?

-----

```
SELECT SUBSTR('YourHome', 1, 3) FROM DUAL;
```

-----

- ① YourHomeYourHomeYourHome
- ② Home
- ③ Your
- ④ You





## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 중 트랜잭션의 일반적인 특징으로 가장 적절하지 않은 것은 무엇인가?

- ① 원자성
- ② 일관성
- ③ 격리성
- ④ 완전성



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 SQL 문의 실행 결과로 가장 적절한 것은 무엇인가?

-----

```
SELECT MOD(-15, -2) FROM DUAL;
```

-----

- ① -1
- ② 1
- ③ NULL
- ④ 0



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음과 같은 조건에서 실행된 SQL의 결과로 가장 적절한 것은 무엇인가?

```
-----  
SELECT employee_id, employee_name, department_id  
FROM employees  
WHERE department_id NOT IN (  
    SELECT department_id  
    FROM departments  
);  
-----
```

※ 보기 구문 생략

Employees Table

| employee_id | employee_name | department_id |
|-------------|---------------|---------------|
| 1           | John Doe      | A             |
| 2           | Jane Smith    | B             |
| 3           | Mike Brown    | C             |

Departments Table

| department_id | department_name |
|---------------|-----------------|
| A             | HR              |
| B             | IT              |

결과 세트

| employee_id | employee_name | department_id |
|-------------|---------------|---------------|
| 3           | Mike Brown    | C             |



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 설명 중 가장 적절하지 않은 것은 무엇인가?

- ① 외래키 : 참조 무결성을 제약한다.
- ② 기본키 : 테이블 당 1개의 기본키를 가짐
- ③ DEFAULT : NULL 값을 가질 수 없음
- ④ CHECK : 입력 값의 범위를 제한



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음과 같은 조건에서 실행된 SQL의 결과로 가장 적절한 것은 무엇인가?

-----

```
SELECT employee_id, employee_name, department_id
```

```
FROM employees
```

```
WHERE department_id IN ('B', NULL);
```

-----

※ 보기 구문 생략

Employees Table

| employee_id | employee_name | department_id |
|-------------|---------------|---------------|
| 1           | John Doe      | A             |
| 2           | Jane Smith    | B             |
| 3           | Mike Brown    | NULL          |

예상 결과

| employee_id | employee_name | department_id |
|-------------|---------------|---------------|
| 2           | Jane Smith    | B             |



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음과 같은 조건에서 실행된 SQL의 결과로 가장 적절한 것은 무엇인가?

```
-----  
SELECT employee_id, employee_name, birth_date  
FROM employees  
WHERE birth_date BETWEEN '1990-01-01' AND '1995-12-31';  
-----
```

※ 보기 구문 생략

Employees Table

| employee_id | employee_name | birth_date |
|-------------|---------------|------------|
| 1           | John Doe      | 1985-05-10 |
| 2           | Jane Smith    | 1990-07-15 |
| 3           | Mike Brown    | 1992-03-01 |

결과 세트

| employee_id | employee_name | birth_date |
|-------------|---------------|------------|
| 2           | Jane Smith    | 1990-07-15 |
| 3           | Mike Brown    | 1992-03-01 |



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음과 같은 조건에서 실행된 SQL의 결과로 가장 적절한 것은 무엇인가?

-----

```
SELECT Year, Month, Product, SUM (Amount) AS TotalSales  
FROM Sales  
GROUP BY ROLLUP (Year, Month, Product);
```

-----

※ 보기 구문 생략

| Year | Month | Product   | Amount |
|------|-------|-----------|--------|
| 2023 | 1     | Product A | 100    |
| 2023 | 1     | Product B | 200    |
| 2023 | 2     | Product A | 150    |
| 2023 | 2     | Product B | 300    |

| Year | Month | Product   | TotalSales |
|------|-------|-----------|------------|
| 2023 | 1     | Product A | 100        |
| 2023 | 1     | Product B | 200        |
| 2023 | 1     | NULL      | 300        |
| 2023 | 2     | Product A | 150        |
| 2023 | 2     | Product B | 300        |
| 2023 | 2     | NULL      | 450        |
| 2023 | NULL  | NULL      | 750        |
| NULL | NULL  | NULL      | 750        |



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) user\_t 테이블의 jumsu 컬럼이 있고, 총 5개의 레코드에 각각 100, 90, 80, NULL, 70 의 데이터가 있다고 가정할 경우  
다음 중 다른 결과를 갖는 SQL 문은 무엇인가?

- ① SELECT COUNT(100) FROM user\_t;
- ② SELECT COUNT(90) FROM user\_t;
- ③ SELECT COUNT(\*) FROM user\_t;
- ④ SELECT COUNT(jumsu) FROM user\_t;





## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음 중 사용자의 권한을 제거(회수)하는 명령어는 무엇인가?

- ① GRANT
- ② REVOKE
- ③ ROLLBACK
- ④ CANCEL



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) 다음과 같은 조건에서 실행된 SQL의 결과로 가장 적절한 것은 무엇인가?

```
-----  
SELECT COUNT(*)  
FROM employees  
WHERE manager_id IS NULL OR department_id IN (101, 103);  
-----
```

- ① 1
- ② 2
- ③ 4
- ④ 0

Employees Table

| employee_id | employee_name | department_id | manager_id |
|-------------|---------------|---------------|------------|
| 1           | John Doe      | 101           | NULL       |
| 2           | Jane Smith    | 102           | 1          |
| 3           | Mike Brown    | NULL          | 2          |
| 4           | Karen White   | 103           | 2          |



## 06. 기출문제 동향과 풀이

### 주제별 문제 풀이

문) SQL 집합 연산자에서 교집합에 해당하는 것은 무엇인가?

- ① INTERSECT
- ② UNION
- ③ EXCEPT
- ④ UNION ALL



**감사합니다**  
THANK YOU