#데이터베이스 #SQL #Join #AdvancedSQL

---

1. 아젠다

- Join
- Views
- Window functions
- Keys

---

2. Join 연산 (Join Operations)

- Join 연산은 두 개의 릴레이션을 입력받아 다른 하나의 릴레이션을 반환합니다.
- Join은 두 릴레이션의 튜플들이 매치되어야 하는 Cartesian product입니다.
    - 또한 Join 결과에 존재할 속성들을 명시합니다 (Projection).
- 일반적으로 FROM clause에서 서브쿼리 표현식으로 사용됩니다.
- **Join 유형 (Join types)**
    - INNER JOIN
    - OUTER JOIN
- **Join 조건 (Join conditions)**
    - NATURAL
    - `[[ON]] <predicate>`
    - `[[USING]] (A1, A2, ..., An)`

---

3. 실행 예제 (Running Example)

3.1. Relations: student, takes

- student relation:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

- takes relation:

ID	course_id	sec_id	semester	year	grade
00128	CS-101	1	Fall	2017	A
00128	CS-347	1	Fall	2017	A-
12345	CS-101	1	Fall	2017	C
12345	CS-190	2	Spring	2017	A
12345	CS-315	1	Spring	2018	A
12345	CS-347	1	Fall	2017	A
19991	HIS-351	1	Spring	2018	B
23121	FIN-201	1	Spring	2018	C+
44553	PHY-101	1	Fall	2017	B-
45678	CS-101	1	Fall	2017	F

ID	course_id	sec_id	semester	year	grade
45678	CS-101	1	Spring	2018	B+
45678	CS-319	1	Spring	2018	B
54321	CS-101	1	Fall	2017	A-
54321	CS-190	2	Spring	2017	B+
55739	MU-199	1	Spring	2018	A-
76543	CS-101	1	Fall	2017	A
76543	CS-319	2	Spring	2018	A
76653	EE-181	1	Spring	2017	C
98765	CS-101	1	Fall	2017	C-
98765	CS-315	1	Spring	2018	B
98988	BIO-101	1	Summer	2017	A
98988	BIO-301	1	Summer	2018	

3.2. Relations: course, instructor

- course relation:

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3

course_id	title	dept_name	credits
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

- instructor relation:

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

---

4. Natural Join

- Natural join은 모든 공통 속성에 대해 동일한 값을 갖는 튜플들을 매칭시키고, 각 공통 열의 사본 하나만 유지합니다.
- 예시: 학생들이 수강한 과목의 ID와 함께 학생들의 이름을 나열합니다.
- FROM clause는 natural join을 사용하여 결합된 여러 릴레이션을 가질 수 있습니다:

```
SELECT A1, A2, ... An
FROM r1 NATURAL JOIN r2 NATURAL JOIN ... NATURAL JOIN rn
```

```
WHERE P;
```

4.1. 주의사항 (Caveat)

> ⚠ Warning
>
> 이름이 같지만 관련 없는 속성들이 잘못 동일시될 수 있음에 주의해야 합니다.

- 예시 (부정확): 다음 쿼리는 `student`, `takes`, `course` 릴레이션 모두에 존재하는 공통 속성(여기서는 `dept_name`)을 기준으로 조인하려고 시도합니다.

```
SELECT dept_name, course_id, name, title, credits
FROM student NATURAL JOIN takes NATURAL JOIN course;
```

  - 이 쿼리는 학생의 학과(`student.dept_name`)와 과목의 학과(`course.dept_name`)가 같은 경우에만 결과를 반환합니다. 즉, 학생이 자신의 학과가 아닌 다른 학과의 과목을 수강한 경우 (예: Comp. Sci. 학생이 History 과목 수강) 해당 (학생 이름, 과목 제목) 쌍이 누락됩니다.
- 예시 (수정된 쿼리): 학생 이름과 수강한 과목의 제목을 나열합니다.
  - 올바른 방법:

```
SELECT name, title
FROM student NATURAL JOIN takes, course
WHERE takes.course_id = course.course_id;
```

  - 부정확한 방법 (Natural Join 사용 시):

```
SELECT name, title
FROM student NATURAL JOIN takes NATURAL JOIN course;
```

> ⚠ Caution

이 쿼리는 학생이 자신의 학과가 아닌 다른 학과의 과목을 수강한 모든 (학생 이름, 과목 제목) 쌍을 생략합니다. (`dept_name` 속성이 `student` 와 `course` 모두에 존재하기 때문)

---

5. Natural Join과 USING 절

- 속성이 잘못 동일시되는 위험을 피하기 위해 `[[USING]]` 구문을 사용합니다.
- `[[USING]]` : 어떤 열이 동일해야 하는지 정확하게 명시할 수 있게 합니다.
- 예시: 학생 이름과 수강한 과목 제목을 나열합니다. (`course_id` 만 조인 기준으로 명시)

```
SELECT name, title
FROM (student NATURAL JOIN takes) JOIN course USING
(course_id);
```

---

6. JOIN ... ON 조건

- `[[ON]]` 조건은 조인되는 릴레이션에 대한 일반적인 술어(predicate)를 허용합니다.
- WHERE clause 술어처럼 작성됩니다.
- 예시:

```
SELECT *
FROM student JOIN takes ON student.ID = takes.ID;
```

  - `[[ON]]` 조건은 `student` 의 튜플과 `takes` 의 튜플의 `ID` 값이 같으면 매치된다고 명시합니다.
- 동등한 쿼리 (WHERE 사용):

```sql
SELECT name, course_id
FROM student, takes
WHERE student.ID = takes.ID;
```

---

7. Inner Join

- Inner join: 매치되지 않는 튜플을 보존하지 않습니다. #JoinType
  - 테이블은 `[[ON]]` 또는 `[[USING]]` 절에 명시된 공통 열을 기반으로 조인됩니다.
  - `[[ON]]` 또는 `[[USING]]` 구문으로 조건을 명시할 수 있습니다.
- 비교: **Natural join**:
  - Natural join은 조인 조건이 두 테이블의 이름이 같은 열이 매치되는 것이라고 가정합니다.
  - `[[ON]]` 또는 `[[USING]]`을 사용할 수 없습니다.
  - Natural join의 결과에서는 반복되는 열이 제거됩니다.
- Natural join 예시 (튜플 손실 가능성): `course`와 `prereq` 테이블을 조인하면, 양쪽 테이블에 모두 존재하지 않는 과목 정보는 손실됩니다.

```sql
SELECT *
FROM course NATURAL JOIN prereq;
```

---

9. Outer Join

- Outer join은 정보 손실을 피하기 위한 join operation의 확장입니다. #JoinType
- Outer join은 조인에서 손실될 수 있는 튜플들을 널(null) 값을 포함하는 튜플을 결과에 생성하여 보존합니다.
- 조인을 계산한 다음, 한 릴레이션에서 다른 릴레이션의 튜플과 매치되지 않는 튜플들을

결과에 추가합니다.

- Outer join의 세 가지 형태:
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN

9.1. Outer Join 실행 예제 (course, prereq)

- Relation course:

course_id	title	dept_name	credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relation prereq:

course_id	prereq_id
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

> ⓘ Info
>
> - `course` 테이블에는 `CS-347`이 없습니다.
> - `prereq` 테이블에는 `CS-315`가 없습니다.

9.2. Inner Join with NATURAL (복습)

- Natural join: 조인되는 릴레이션 중 하나 또는 둘 다에 있는 일부 튜플이 손실될 수 있습니다.

```
SELECT *
FROM course NATURAL JOIN prereq;
```

- 결과:

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101

9.3. Left Outer Join with NATURAL

- Left outer join: 연산자 앞(왼쪽)에 명명된 릴레이션의 튜플만 보존합니다.

```
SELECT *
FROM course NATURAL LEFT OUTER JOIN prereq;
```

- 결과: (course의 모든 튜플 포함, 매칭되는 prereq 없으면 null)

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	`<null>`

9.4. Right Outer Join with NATURAL

- Right outer join: 연산자 뒤(오른쪽)에 명명된 릴레이션의 튜플만 보존합니다.

```
SELECT *
FROM course NATURAL RIGHT OUTER JOIN prereq;
```

- 결과: (prereq의 모든 튜플 포함, 매칭되는 course 없으면 null)

course_id	prereq_id	title	dept_name	credits
BIO-301	BIO-101	Genetics	Biology	4
CS-190	CS-101	Game Design	Comp. Sci.	4
CS-347	CS-101	`<null>`	`<null>`	`<null>`

9.5. Full Outer Join with NATURAL

- Full outer join: 양쪽 릴레이션의 모든 튜플을 보존합니다. 매칭되지 않는 속성은 null로 채워집니다.

```
SELECT *
FROM course NATURAL FULL OUTER JOIN prereq;
```

  - 결과:

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	`<null>`
CS-347	`<null>`	`<null>`	`<null>`	CS-101

> ⚠ **Warning**
>
> MySQL은 `[[FULL OUTER JOIN]]`을 지원하지 않습니다. #MySQL #Compatibility

- 대안: Left outer join과 Right outer join의 UNION 사용

```
SELECT course_id, title, dept_name, credits, prereq_id
FROM course NATURAL LEFT OUTER JOIN prereq
UNION
```

```sql
SELECT course_id, title, dept_name, credits, prereq_id
FROM course NATURAL RIGHT OUTER JOIN prereq;
```

> **Important**
> UNION을 제대로 수행하려면 두 조인 쿼리의 속성이 정렬되어야 합니다 (순서와 타입이 일치해야 함).

---

10. Outer Join 예제 (Student, StudentCourse)

- 테이블: (섹션 8 참고)
  - Student
  - StudentCourse
- Left Join 예제 (`ON` 사용):

```sql
SELECT Student.NAME, StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse -- LEFT OUTER JOIN과 동일
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

  - 결과: (Student의 모든 행 포함)

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	`<null>`
ROHIT	`<null>`

NAME	COURSE_ID
NIRAJ	<null>

- Right Join 예제 (`ON` 사용):

```
SELECT Student.NAME, StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse -- RIGHT OUTER JOIN과 동일
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

  - 결과: (StudentCourse의 모든 행 포함)

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
<null>	4
<null>	5
<null>	4

- Full Join 예제 (`ON` 사용):

```
SELECT Student.NAME, StudentCourse.COURSE_ID
FROM Student
FULL JOIN StudentCourse -- FULL OUTER JOIN과 동일
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

  - 결과: (양쪽 테이블의 모든 행 포함)

NAME	COURSE_ID
HARSH	1

NAME	COURSE_ID
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	`<null>`
ROHIT	`<null>`
NIRAJ	`<null>`
`<null>`	9
`<null>`	10
`<null>`	11

---

11. Join 유형 및 조건 요약

- 조인 유형: 다른 릴레이션의 어떤 튜플과도 일치하지 않는 각 릴레이션의 튜플을 어떻게 처리할지 정의합니다. #JoinType
  - INNER JOIN: 매치되는 튜플만 반환.
  - LEFT OUTER JOIN: 왼쪽 테이블의 모든 튜플 + 매치되는 오른쪽 튜플 (없으면 null).
  - RIGHT OUTER JOIN: 오른쪽 테이블의 모든 튜플 + 매치되는 왼쪽 튜플 (없으면 null).
  - FULL OUTER JOIN: 양쪽 테이블의 모든 튜플 (매치 안되면 null).
- 조인 조건: 두 릴레이션에서 어떤 튜플이 일치하는지 정의합니다. #JoinCondition
  - NATURAL: 동일한 속성 이름과 데이터 타입을 기반으로 조인.
  - ON `<predicate>` : ON 절에 명시적으로 지정된 열(들) 또는 조건 기반으로 조인.
  - USING `(A1, A2, ..., An)` : USING 다음에 나열된 공통 속성 이름(들) 기반으로 조인.

11.1. SQL Joins 시각화 (개념적)

(슬라이드 26의 다이어그램 설명)

- INNER JOIN: 두 테이블(A, B)의 교집합. (A.key = B.key)
- LEFT OUTER JOIN: 테이블 A 전체 + 테이블 B의 교집합 부분. (A의 모든 행 + B의 매칭 행, 매칭 안되면 B 컬럼은 null)
- RIGHT OUTER JOIN: 테이블 B 전체 + 테이블 A의 교집합 부분. (B의 모든 행 + A의 매칭 행, 매칭 안되면 A 컬럼은 null)
- FULL OUTER JOIN: 테이블 A와 B의 합집합. (양쪽 모든 행, 매칭 안되면 해당 테이블 컬럼은 null)
- LEFT ANTI JOIN (개념): 테이블 A에는 있지만 테이블 B에는 없는 부분. (SQL: `LEFT JOIN ... WHERE B.key IS NULL`)
- RIGHT ANTI JOIN (개념): 테이블 B에는 있지만 테이블 A에는 없는 부분. (SQL: `RIGHT JOIN ... WHERE A.key IS NULL`)
- FULL ANTI JOIN (개념): 테이블 A 또는 B에만 있는 부분 (교집합 제외). (SQL: `FULL OUTER JOIN ... WHERE A.key IS NULL OR B.key IS NULL`)

---

12. Join 조건 상세

- NATURAL: 동일한 속성 이름과 데이터 타입을 기반으로 두 테이블을 조인합니다.

```
SELECT * FROM course NATURAL JOIN prereq;
```

- ON `<predicate>` : ON 절에 명시적으로 지정된 열(들) 또는 조건을 기반으로 두 테이블을 조인합니다.

```
SELECT * FROM course JOIN prereq ON course.course_id =
prereq.prereq_id; -- 주의: 이 예제는 prereq 테이블 구조상 의미가 다를
수 있음. 슬라이드 예제에서는 course_id로 연결
-- 슬라이드 원본 예시 반영:
```

```
SELECT * FROM course JOIN prereq ON course.course_id =
prereq.course_id;
```

- USING `(A1, A2, ..., An)` : USING 다음에 나열된 공통 속성 이름(들)을 기반으로 두 테이블을 조인합니다.

```
SELECT * FROM course JOIN prereq USING (course_id);
```

---

13. Inner Join vs. Natural Join 비교

특징	INNER JOIN	NATURAL JOIN
조인 기준	`[[ON]]` 절에 명시적으로 지정된 열 또는 조건	동일한 [[attribute
결과 테이블 속성	두 테이블의 모든 속성을 포함 (중복 열 포함 가능)	두 테이블의 모든 속성을 포함하되, 각 공통 열의 사본 하나만 유지
반환 레코드	두 테이블 모두에 존재하는 레코드만 반환	`LEFT`, `RIGHT`, `FULL` 표시가 없으면 공통 열을 기반으로 행을 반환

- Inner join 구문:

```
SELECT * FROM course
INNER JOIN prereq ON course.course_id = prereq.course_id;
```

  - 이것은 다음과 동등합니다:

```
SELECT * FROM course
JOIN prereq ON course.course_id = prereq.course_id;
```

- Natural join 구문:

```sql
SELECT *
FROM course NATURAL JOIN prereq;
```

> ⚠ **Warning**
> NATURAL JOIN과 ON 조건을 함께 사용하는 것은 유효하지 않습니다.
>
> ```sql
> -- 잘못된 구문
> SELECT *
> FROM course NATURAL JOIN prereq
> ON course.course_id = prereq.prereq_id; -- NOT VALID!
> ```

- 결과 비교 (Inner Join vs Natural Join 예제):
  - Inner Join (`ON course.course_id = prereq.course_id`):

course.course_id	course.title	course.dept_name	course.credits	p
BIO-301	Genetics	Biology	4	B
CS-190	Game Design	Comp. Sci.	4	C

  - Natural Join:

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101

(Natural Join은 공통 열 `course_id` 를 하나만 표시)

---

14. Outer Join vs. Natural Join 비교

14.1. Right Outer Join

- `NATURAL RIGHT OUTER JOIN`:

```
SELECT *
FROM course NATURAL RIGHT OUTER JOIN prereq;
```

  - 결과:

course_id	prereq_id	title	dept_name	credits
BIO-301	BIO-101	Genetics	Biology	4
CS-190	CS-101	Game Design	Comp. Sci.	4
CS-347	CS-101	`<null>`	`<null>`	`<null>`

- `RIGHT OUTER JOIN ... USING` (동등):

```
SELECT *
FROM course RIGHT OUTER JOIN prereq USING (course_id);
```

(결과는 위와 동일)

- `RIGHT OUTER JOIN ... ON`:

```
SELECT *
FROM course RIGHT OUTER JOIN prereq ON course.course_id =
prereq.course_id;
```

  - 결과 (열 이름 유지):

course.course_id	course.title	course.dept_name	course.credits	p
BIO-301	Genetics	Biology	4	B
CS-190	Game Design	Comp. Sci.	4	C
`<null>`	`<null>`	`<null>`	`<null>`	C

14.2. Left Outer Join

- `NATURAL LEFT OUTER JOIN`:

```
SELECT *
FROM course NATURAL LEFT OUTER JOIN prereq;
```

  - 결과:

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	`<null>`

- `LEFT OUTER JOIN ... USING` (동등):

```
SELECT *
FROM course LEFT OUTER JOIN prereq USING (course_id);
```

(결과는 위와 동일)

- `LEFT OUTER JOIN ... ON`:

```
SELECT *
FROM course LEFT OUTER JOIN prereq ON course.course_id =
prereq.course_id;
```

  - 결과 (열 이름 유지):

course.course_id	course.title	course.dept_name	course.credits	p
BIO-301	Genetics	Biology	4	B
CS-190	Game Design	Comp. Sci.	4	C
CS-315	Robotics	Comp. Sci.	3	`<`

---

15. Natural Join은 종종 회피됨

> ⚠ **Caution**
>
> Natural join은 실무에서 종종 회피됩니다. 이유는 다음과 같습니다:

- Natural join은 (대부분의 SQL 코더에게) 가독성이 특별히 좋지 않고, 다양한 도구/라이브러리에서 지원되지 않을 수 있습니다.
- Natural join은 정보 제공적이지 않습니다; 스키마를 참조하지 않고는 어떤 열이 조인되는지 알 수 없습니다.
- 조인 조건이 보이지 않게 스키마 변경에 취약합니다.
  - 여러 개의 natural join 열이 있고 그 중 하나가 테이블에서 제거되더라도 쿼리는 여전히 실행됩니다.
  - 하지만 결과가 정확하지 않을 수 있으며 이러한 동작 변경은 조용히(silent) 발생합니다.
- 노력할 가치가 거의 없습니다; 특정 조건을 입력하지 않음으로써 약 10초 정도만 절약할 뿐입니다.

---

16. 다음 내용 (Coming next)

- Advanced SQL (계속)
  - Views
  - Window functions
  - Keys

---

17. 핵심 주요 키워드

- Join
- Views

- Window functions
- Keys
- 릴레이션
- 튜플
- Cartesian product
- 속성
- Projection
- FROM clause
- 서브쿼리
- 조인 유형
- INNER JOIN
- OUTER JOIN
- 조인 조건
- NATURAL
- ON
- USING
- Natural join
- WHERE clause
- SELECT
- null values
- Left outer join
- Right outer join
- Full outer join
- Inner join
- MySQL
- Compatibility
- UNION
- Left Join
- Right Join
- Full Join
- 조인 유형

- 조인 조건
- SQL
- 스키마
- 스키마 변경