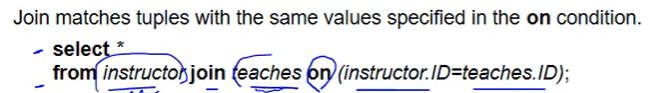
**Chapter 04– Intermediate SQL**

**- Join Expressions**

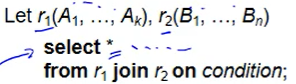
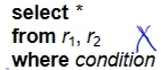
Join operation: 두 relation을 받아서 하나의 결과를 내 놓음. 특정한 조건에서 두 relation이 매칭되는 튜플들에 대한 Cartesian product. 또한 결과의 attribute도 특정할 수 있음.

SQL에서는 from에서 join 사용 가능.

**- Join Operation**



on으로 조건 사용.

 ⬄ 둘이 같음.

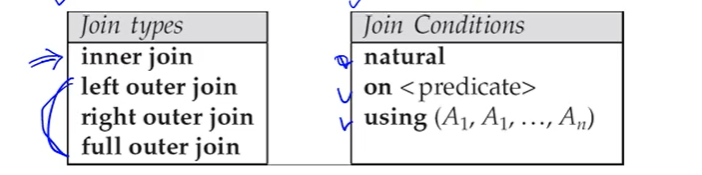
From 절에 Expression을 사용할 수 있음.



**- Joined Relations**

Join condition과 Join type이 있음.

**natural은 앞에 붙어서 type으로 착각할 수도 있지만 type이 아니라 condition임.**

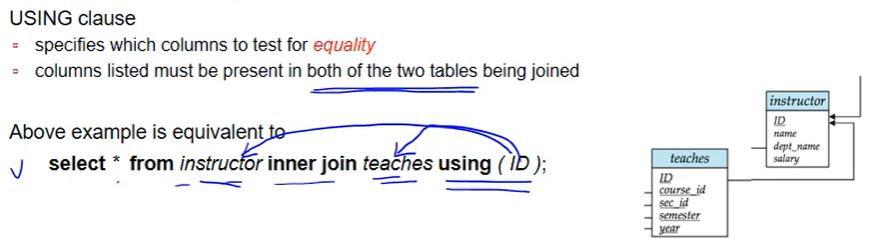


**- Inner Join**

이건 그냥 join과 같음. Default. 즉 그냥 join만 쓰면 inner join이라는 것.



‘On’ clause.

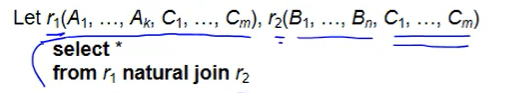


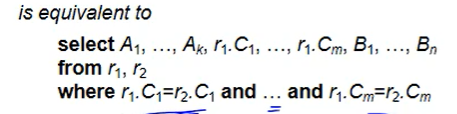
using은 같은 이름을 가진 attribute 나열하는 방식으로 사용.

위에서 양 쪽의 ID가 같은 조합만 결과를 냄. 두 테이블에 공통된 attribute가 다른 것이 있어도 USING에 있는 것만 사용.

**- Natural Join**

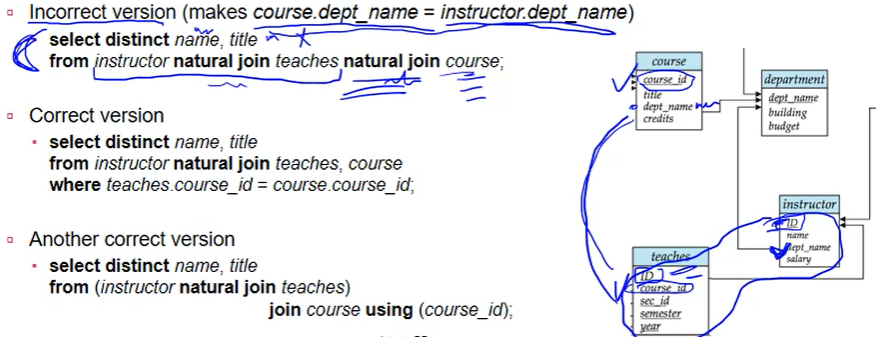
Natural join은 **모든 공통된 attribute**에 대해 매칭.

 이것과

 이건 같음.

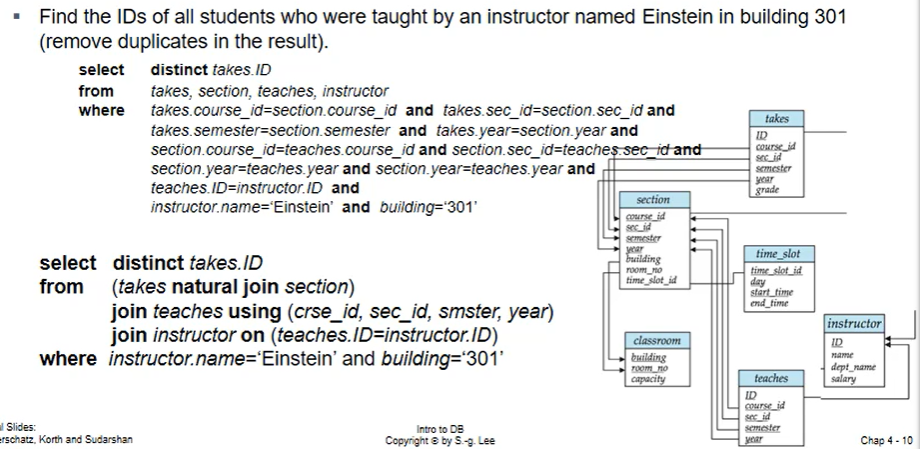
즉 select \* from r1 join r2 using (C1, C2, …, Cm)과 같음.

**Natural join의 위험성: 서로 관계 없는 attribute가 이름이 같으면 계산이 이상하게 됨.**



위와 같이 where condition을 쓰거나 using을 사용해줘야 함. 왜냐면 instructor에도 dept\_name이 있고 course에도 dept\_name이 있기 때문에 이거 두 개도 같이 조인해버림.

근데 이걸 쓰는 이유.. 복잡한 where문을 간단하게 만들어 줆.



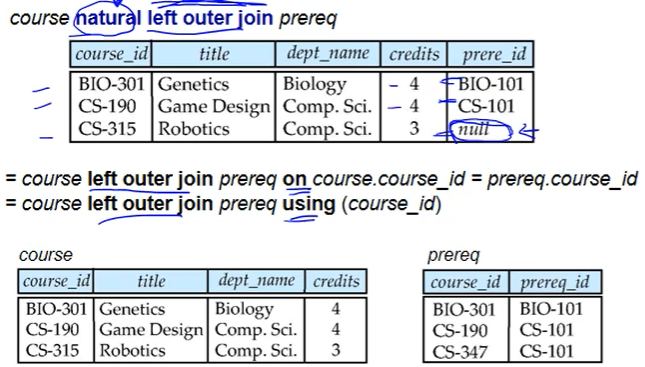
위와 같이 간단하게 나타나짐!

**- Outer Join**

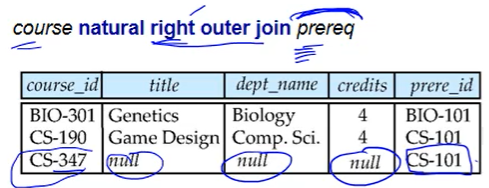
정보의 손실을 피하기 위해 사용하는 join operation.

Join을 계산한 뒤 매칭되지 않는 tuple을 null value를 사용해 붙여줌.

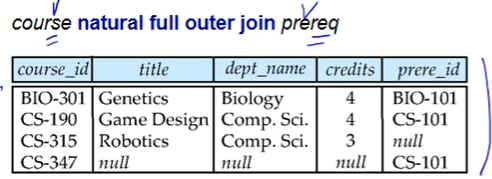
**- Left Outer Join**



**- Right Outer Join**



**- Full Outer Join**



**- Integrity Constraints (무결성 조건)**

무결성 조건은 데이터베이스에 accidental 데미지를 주는 것을 방지한다.

**Not null, unique, primary key, check(P)** 등을 사용.

**- Not Null Constraint**

Attribute가 Null 이 되면 안 된다는 조건. Update나 insert가 일어날 때 체크.

얘는 create 문에서 attribute, type 바로 뒤에 붙임.

**- The Unique & Primary Key Constraint**

Unique: attribute가 유일하다는 것을 보장. Null이 될 수도 있음. Unique(a, b)면 a와 b의 조합이 유일하다는 의미. 따로따로가 아님. 후보키는 유일해야 하므로 후보키가 된다.

Primary Key: Unique & Not Null.

이건 밑에서 Unique(a, b, c) 같은 식으로 선언. Primary key도 마찬가지.

단 Primary key가 1개일 경우에는 not null처럼 attribute 옆에다 쓸 수도 있음.

**- The check Clause**

Check(p) 에서 P는 predicate임.



이런 식으로 attribute 값의 범위를 한정. 이 것도 insert, update 때 체크.

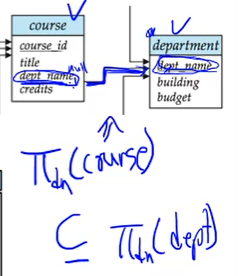
**- Referential Integrity == Foreign Key Constraint (참조 무결성)**

하나의 attribute의 set은 다른 relation의 attribute의 set에도 나타나야 한다.

이것도 Create table을 할 때 선언됨.

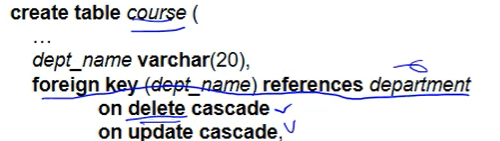
Foreign key: foreign key를 이루고 있는 attribute와 참조하는 relation 이름을 씀.

Foreign key (a, b, c, d) references T. <= 이런 식으로.

 department에서 delete하거나 course에서 insert, update를 할 때 체크함.

Create table에 포함되고, 기본으로 foreign key는 참조하는 table의 primary key를 참조한다.

**- Cascading Actions in Referential Integrity**



참조를 위반할 때 참조하는 (혹은 참조되는) record들도 변경되게 해줄 수 있음.

**On delete: delete에서 참조 위반하면 참조하는 record들도 다 delete 하라는 의미.**

**On Update도 똑같은 의미.**

다른 액션들도 있음: set null, set default. Ex) on delete set null, on update set default.

**- Authorization Specification in SQL**

권한은 DBA한테서 받음. Grant 명령문은 명령을 주기 위해 사용.

문법: grant <privilege list> on <relation name or view name> to <user list>

**- Privileges in SQL**

Select, insert, update, delete, all privileges가 있음.

**- Revoking Authorization in SQL**

권한을 다시 가져오는 것.

문법: revoke <privilege list> on <relation name or view name> from <user list>

**- Authorization-Grant Graph**

DBA가 유저들에게 권한을 주면, 권한을 받은 유저들도 다른 유저들에게 권한을 전달할 수 있다.

이 때 DBA가 권한을 revoke하면 그 유저가 권한을 전달한 유저들의 권한도 없어짐.

**- Roles**

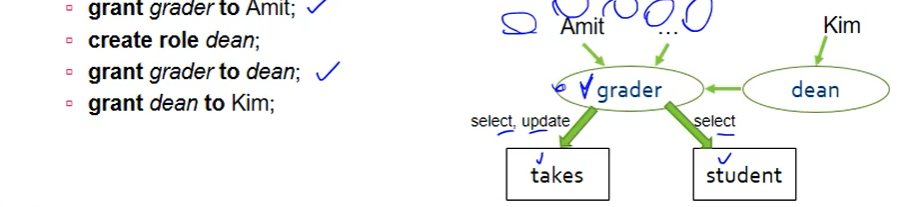
Role은 유저와 그 권한의 그룹을 표현하기 위해 사용.

문법: create role ROLE\_NAME;

Role에 권한을 줄 수 있음.

Ex: grant select, update on takes to grader

Role 또한 다른 유저나 다른 role에게 넘겨질 수 있음.



Authorization은 DBMS dependent함. DBMS마다 문법이나 규칙 등이 다른 듯.

기본적으로 grant, revoke, role만 알면 됨.