ECE30030/ITP30010 Database Systems

Advanced SQL

Reading: Chapters 4-5

Charmgil Hong

charmgil@handong.edu

Spring, 2025

Handong Global University

Agenda

- Join
- Views
- Window functions
- Keys

Join Operations

- Join operations take two relations and return another relation
  - A join is a Cartesian product that requires tuples in the two relations match
    - It also specifies the attributes that are present in the result of the join (project)
  - Typically used as subquery expressions in the **FROM** clause

  - Join types
    - **INNER JOIN**
    - **OUTER JOIN**

  - Join conditions
    - **NATURAL**
    - **ON** <predicate>
    - **USING (**A_1, A_2, …, A_n**)**

Running Example

- Relations: student, takes

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

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
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	*<null>*

Running Example

- Relations: course, instructor

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
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

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

Natural Join

- Natural join matches tuples with the same values for all common attributes, and retains only one copy of each common column
  - *E.g.*, List the names of students along with the course ID of the courses that they took
    - **SELECT** *name*, *course_id*
      FROM *student, takes*
      WHERE *student.ID = takes.ID*;
  - Same query in SQL with natural join:
    - **SELECT** *name*, *course_id*
      FROM *student* **NATURAL JOIN** *takes*;

name	course_id
Zhang	CS-101
Zhang	CS-347
Shankar	CS-101
Shankar	CS-190
Shankar	CS-315
Shankar	CS-347
Brandt	HIS-351
Chavez	FIN-201
Peltier	PHY-101
Levy	CS-101
Levy	CS-101
Levy	CS-319
Williams	CS-101
Williams	CS-190
Sanchez	MU-199
Brown	CS-101
Brown	CS-319
Aoi	EE-181
Bourikas	CS-101
Bourikas	CS-315
Tanaka	BIO-101
Tanaka	BIO-301

Natural Join

- The **FROM** clause can have multiple relations combined using natural join:
  - **SELECT** $A_1, A_2, \ldots A_n$
    FROM r_1 **NATURAL JOIN** r_2 **NATURAL JOIN** … **NATURAL JOIN** r_n
    WHERE P ;

Caveat

- *E.g.*, **(Incorrect)**
  SELECT *dept_name, course_id, name, title, credits*
  FROM *student* **NATURAL JOIN** *takes* **NATURAL JOIN** *course*;

dept_name	course_id	name	title	credits
Biology	BIO-101	Tanaka	Intro. to Biology	4
Biology	BIO-301	Tanaka	Genetics	4
Comp. Sci.	CS-101	Zhang	Intro. to Computer Science	4
Comp. Sci.	CS-101	Shankar	Intro. to Computer Science	4
Comp. Sci.	CS-101	Williams	Intro. to Computer Science	4
Comp. Sci.	CS-101	Brown	Intro. to Computer Science	4
Comp. Sci.	CS-190	Shankar	Game Design	4
Comp. Sci.	CS-190	Williams	Game Design	4
Comp. Sci.	CS-315	Shankar	Robotics	3
Comp. Sci.	CS-319	Brown	Image Processing	3
Comp. Sci.	CS-347	Zhang	Database System Concepts	3
Comp. Sci.	CS-347	Shankar	Database System Concepts	3
Elec. Eng.	EE-181	Aoi	Intro. to Digital Systems	3
Finance	FIN-201	Chavez	Investment Banking	3
History	HIS-351	Brandt	World History	3
Music	MU-199	Sanchez	Music Video Production	3
Physics	PHY-101	Peltier	Physical Principles	4

Caveat

- Beware of unrelated attributes with same name getting equated incorrectly
  - *E.g.*, List the names of students along with the titles of courses that they have taken
    - Correct

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

    - Incorrect

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

      - This query omits all (student name, course title) pairs where the student takes a course in a department other than the student's own department

Natural Join with USING Clause

- To avoid the danger of equating attributes erroneously, use the **USING** construct
  - USING: allows us to specify exactly which columns should be equated
  - *E.g.*, **SELECT** *name*, *title*
    FROM (*student* **NATURAL JOIN** *takes*) **JOIN** *course* **USING (***course_id***)**

name	title
Tanaka	Intro. to Biology
Tanaka	Genetics
Zhang	Intro. to Computer Science
Shankar	Intro. to Computer Science
Levy	Intro. to Computer Science
Williams	Intro. to Computer Science
Brown	Intro. to Computer Science
Bourikas	Intro. to Computer Science
Levy	Intro. to Computer Science
Shankar	Game Design
Williams	Game Design
Shankar	Robotics
Bourikas	Robotics
Levy	Image Processing
Brown	Image Processing
Zhang	Database System Concepts
Shankar	Database System Concepts
Aoi	Intro. to Digital Systems
Chavez	Investment Banking
Brandt	World History
Sanchez	Music Video Production
Peltier	Physical Principles

JOIN … ON

- The **ON** condition allows a general predicate over the relations being joined
  - Written like a **WHERE** clause predicate
  - *E.g.*, **SELECT** *
    FROM *student* **JOIN** *takes* **ON** *student.ID = takes.ID*
    - The **ON** condition specifies that a tuple from *student* matches a tuple from *takes* if their *ID* values are equal
    - Equivalent to:
      SELECT *name*, *course_id*
      FROM *student, takes*
      WHERE *student.ID = takes.ID*;

Inner Join

- Inner join: Does not preserve nonmatched tuples
  - Tables are joined based on common columns mentioned in the **ON** or **USING** clause
  - One can specify the condition with an **ON** or **USING** construct

- *C.f.*, Natural join: assumes the join condition to be where same-named columns in both tables match
  - One cannot use **ON** or **USING**
  - In the result of a natural join, repeated columns are avoided

Natural Join

- Natural join: Some tuples in either or both relations being joined may be lost

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

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

Examples

- Tables

ROLL_NO	NAME
1	HARSH
2	PRATIK
3	RIYANKA
4	DEEP
5	SAPTARHI
6	DHANRAJ
7	ROHIT
8	NIRAJ

Student

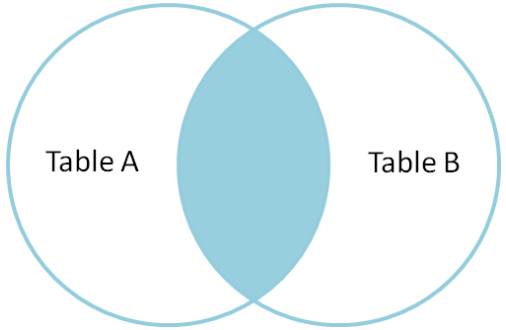
COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

StudentCourse

Examples

- Inner join
  - **SELECT** *StudentCourse.COURSE_ID, Student.NAME*
    FROM *Student*
    INNER JOIN *StudentCourse*
    ON *Student.ROLL_NO = StudentCourse.ROLL_NO;*

COURSE_ID	NAME
1	HARSH
2	PRATIK
2	RIYANKA
3	DEEP
1	SAPTARHI



Outer Join

- An extension of the join operation that avoids loss of information
  - Outer join preserves those tuples that would be lost in a join by creating tuples in the result containing null values
  - Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join

- Three forms of outer join:
  - **LEFT OUTER JOIN**
  - **RIGHT OUTER JOIN**
  - **FULL OUTER JOIN**

Running Example

- 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

  - *course* is missing CS-347
  - *prereq* is missing CS-315

Inner Join with NATURAL

- Natural join: Some tuples in either or both relations being joined may be lost

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

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

Left Outer Join with NATURAL

- Left outer join: Preserves tuples only in the relation named before (to the left of) the operation

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

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*

Right Outer Join with NATURAL

- Right outer join: Preserves tuples only in the relation named after (to the right of) the operation

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

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>*

Full Outer Join with NATURAL

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

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

  - MySQL does NOT support FULL join
    - Alternative: use the **UNION** of left and right joins
      SELECT course_id, title, dept_name, credits, prereq_id
      FROM course **NATURAL LEFT OUTER JOIN** prereq
      UNION
      SELECT course_id, title, dept_name, credits, prereq_id
      FROM course **NATURAL RIGHT OUTER JOIN** prereq;
      - In order to perform UNION properly, the attributes of both join queries must be aligned

Examples

- Tables

ROLL_NO	NAME
1	HARSH
2	PRATIK
3	RIYANKA
4	DEEP
5	SAPTARHI
6	DHANRAJ
7	ROHIT
8	NIRAJ

Student

COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

StudentCourse

Examples

- Left join
  - **SELECT** *Student.NAME, StudentCourse.COURSE_ID*
    FROM *Student*
    LEFT JOIN *StudentCourse*
    ON *StudentCourse.ROLL_NO = Student.ROLL_NO;*

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*

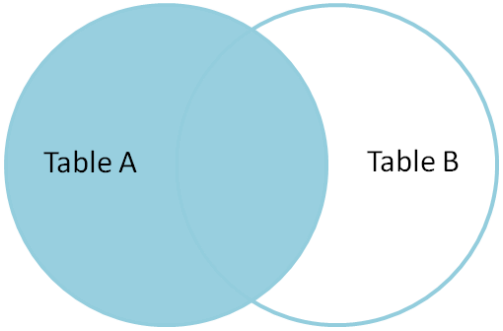
- Right join
  - **SELECT** *Student.NAME, StudentCourse.COURSE_ID*
    FROM *Student*
    RIGHT JOIN *StudentCourse*
    ON *StudentCourse.ROLL_NO = Student.ROLL_NO;*

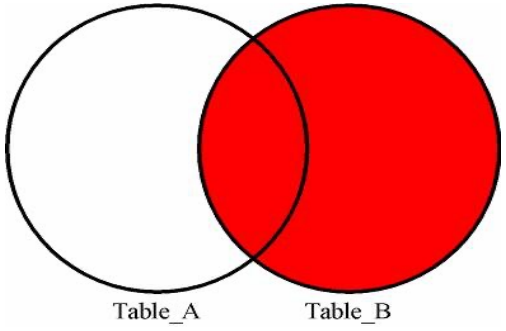
NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
NULL	4
NULL	5
NULL	4

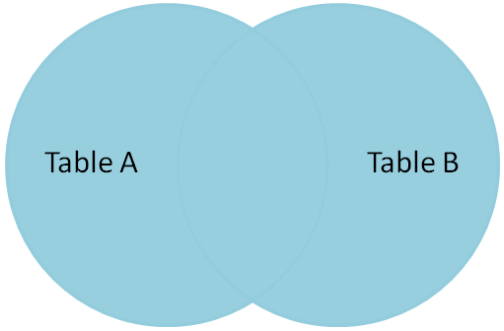
Examples

- Full join
  - **SELECT** *Student.NAME, StudentCourse.COURSE_ID*
    FROM *Student*
    FULL JOIN *StudentCourse*
    ON *StudentCourse.ROLL_NO = Student.ROLL_NO;*

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*
NULL	9
NULL	10
NULL	11





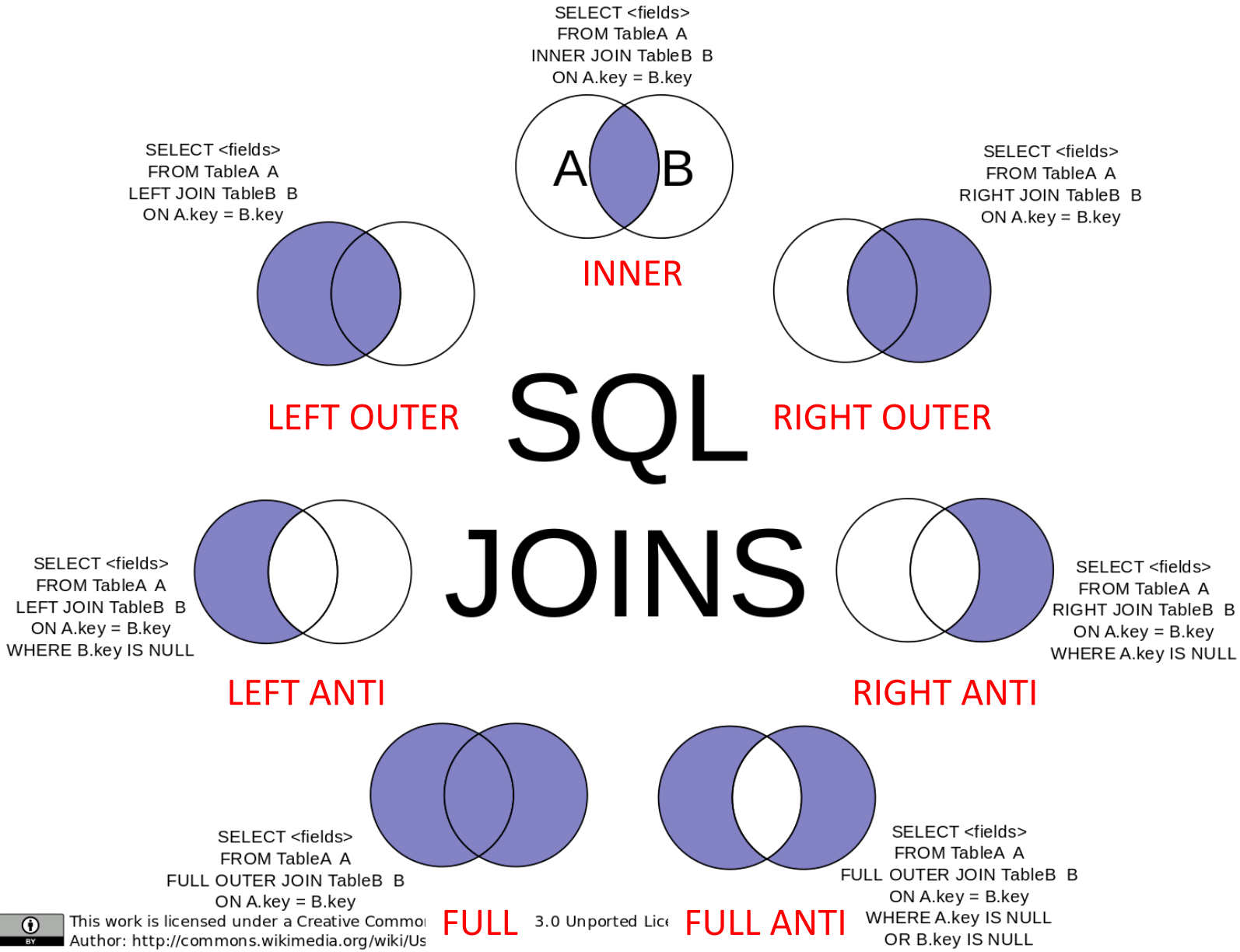


Join Types and Conditions

- Join type: Defines how tuples in each relation that do not match any tuples in the other relation are treated
  - **INNER JOIN**
  - **LEFT OUTER JOIN**
  - **RIGHT OUTER JOIN**
  - **FULL OUTER JOIN**

- Join condition: Defines which tuples in the two relations match
  - **NATURAL**
  - **ON** <predicate>
  - **USING (**A_1, A_2, …, A_n**)**

Join Types



Join Condition

- Join condition
  - **NATURAL**: Joins two tables based on same attribute name and datatypes
    - **SELECT** * **FROM** *course* **NATURAL JOIN** *prereq;*
  - **ON** <predicate>: Joins two tables based on the column(s) explicitly specified in the **ON** clause
    - **SELECT** * **FROM** *course*
      JOIN *prereq* **ON** *course.course_id = prereq.prereq_id*;
  - **USING (**A_1, A_2, …, A_n**)**: Joins two tables based on common attribute name(s) listed next to **USING**
    - **SELECT** * **FROM** *course*
      JOIN *prereq* **USING (***course_id***)**

Inner Join vs. Natural Join

INNER JOIN	NATURAL JOIN
Joins two tables on the basis of the column which is explicitly specified in the ON clause	Joins two tables based on same attribute name and datatypes
The resulting table will contain all the attribute of both the tables (including duplicate columns)	The resulting table will contain all the attribute of both the tables but keep only one copy of each common column
Only those records will return which exists in both tables	If there is no indication of LEFT, RIGHT, or FULL, it returns the rows based on the common column

* Source: https://www.geeksforgeeks.org/difference-between-natural-join-and-inner-join-in-sql/

Inner Join vs. Natural Join

- Inner join
  - **SELECT** * **FROM** *course*
    INNER JOIN *prereq* **ON** *course.course_id = prereq.prereq_id*;

- Natural join
  - **SELECT** *
    FROM *course* **NATURAL JOIN** *prereq*
    ON *course.course_id = prereq.prereq_id*; ← NOT VALID!

Inner Join vs. Natural Join

- Inner join
  - **SELECT** * **FROM** *course*
    INNER JOIN *prereq* **ON** *course.course_id = prereq.course_id*;
    - Equivalent to:
      SELECT * **FROM** *course*
      JOIN *prereq* **ON** *course.course_id = prereq.course_id*;

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

- Natural join
  - **SELECT** *
    FROM *course* **NATURAL JOIN** *prereq*;

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

Outer Join vs. Natural Join

- Right outer join
  - **SELECT** *
    FROM *course* **NATURAL RIGHT OUTER JOIN** *prereq*;
    - Equivalent to:
      SELECT *
      FROM *course* **RIGHT OUTER JOIN** *prereq*
      USING (*course_id*);

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*

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

course.course_id	title	dept_name	credits	prereq.course_id	prereq_id
BIO-301	Genetics	Biology	4	BIO-301	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-190	CS-101
null	*null*	*null*	*null*	CS-347	CS-101

Outer Join vs. Natural Join

- Left outer join
  - **SELECT** *
    FROM *course* **NATURAL LEFT OUTER JOIN** *prereq*;
    - Equivalent to:
      SELECT *
      FROM *course* **LEFT OUTER JOIN** *prereq*
      USING (*course_id*);

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*

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

course.course_id	title	dept_name	credits	prereq.course_id	prereq_id
BIO-301	Genetics	Biology	4	BIO-301	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-190	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*	*null*

Natural Joins Are Often Avoided

- *Natural joins are often avoided in practice*, because:
  - Natural joins are not particularly readable (by most SQL coders) and possibly not supported by various tools/libraries
  - Natural joins are not informative; you cannot tell what columns are being joined on without referring to the schema
  - Your join conditions are invisibly vulnerable to schema changes
    - Even if there are multiple natural join columns and one such column is removed from a table, the query will still execute
    - But the result may not be correct and this change in behavior will be silent
  - Hardly worth the effort; you are only saving about 10 seconds by not typing specific conditions

EOF

- Coming next:
  - Advanced SQL (cont'd)
    - Views
    - Window functions
    - Keys