



## Joins in ANSI/SQL92

- State the **join criteria** in the **WHERE** clause
- Must prefix (qualify) the column name with the table name when the same column name appears in more than one of the joined tables

SQL92

```
SELECT ln, fn, title, type
FROM writer, article
WHERE writer.writerid = article.writerid;
```

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

3

## Inner Join (aka Equijoin)

- A join in which records from two tables are combined and added to the result set only when there are **matching values** in the common fields

SQL92

```
SELECT ln, fn, title, type
FROM writer, article
WHERE writer.writerid = article.writerid;
```

- If a record in one table doesn't have a matching record in the other table, it won't be selected
  - An **Outer Join** will show **all** of a table's records, even if there are no matching records in the other table
    - Chapter 10

4

## Table Ambiguities

- Why do we obtain an error message here?

```
SELECT writerid, ln, fn, title, type
FROM writer, article
WHERE writerid = writerid;
```

```
ORA-00918: column ambiguously defined
00918. 00000 - "column ambiguously defined"
*Cause:
*Action:
Error at Line: 7 Column: 18
```

- Eliminate ambiguity by citing the table name when a field appears in more than one of the joined tables

```
SELECT ln, fn, title, type
FROM writer, article
WHERE writer.writerid = article.writerid;
```

	LN	FN	TITLE
1	Yamamura Shinjiro		The Supreme Court's
2	Waldeck Kristine		Wages and Prices Ho
3	Waldeck Kristine		U.S. Plans Gas Rati
4	Cox Kelly		Milton Friedman Inte
5	Johnson Leroy W.		Trans-Alaskan Oil P
6	Johnson Leroy W.		Farm Aid Abuses

## ANSI/SQL92 Join Practice Time

- Display title, length, and description for any article longer than 2000 words.

## Table Alias

- Makes it easier to refer to a table
- Required when need to join a table to itself
  - Recursive relationship (Chapter 10)

```
SELECT writer.writerid, ln, fn, title, type
   FROM writer, article
   WHERE writer.writerid = article.writerid;
```

```
SELECT w.writerid, ln, fn, title, type
   FROM writer w, article a
   WHERE w.writerid = a.writerid;
```

7

## Nulls and Joins

- NULL is not equal to any other value, including another NULL
- Records having NULL in the join criteria field won't appear in the result set of an inner join
  - an outer join will include them (chapter 10)

8

## Inner Joins: ANSI/SQL99 Syntax (≥9i)

- Alternate syntax to produce an inner join (equijoin)
- Produces same result set as ANSI/SQL92 style join

SQL92

```
SELECT ln, fn, title, type
  FROM writer, article
 WHERE writer.writerid = article.writerid;
```

- **INNER JOIN... ON** syntax
  - ON clause cites fully qualified primary key/foreign key references

SQL99

```
SELECT ln, fn, title, type
  FROM writer INNER JOIN article
         ON writer.writerid = article.writerid;
```

9

## Inner Joins: ANSI/SQL99 Syntax (new in 9i)

- **INNER JOIN... USING** Syntax
  - when primary key and foreign key columns have same name

SQL92

```
SELECT ln, fn, title, type
  FROM writer, article
 WHERE writer.writerid = article.writerid;
```

SQL99

```
SELECT ln, fn, title, type
  FROM writer INNER JOIN article USING(writerid);
```

10

## ANSI/SQL99 Join Practice Time

- Display title, length, and description for any article longer than 2000 words.

11

## Cartesian Product

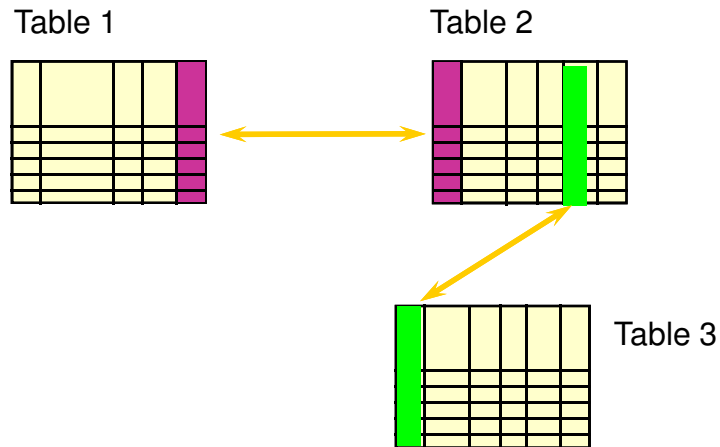
- A result set that joins **every** record in one table to **every** record in another table

```
SQL92 SELECT ln, fn, title, type  
FROM writer, article;
```

- believe it or not, this can be useful...
- Occurs when a join expression is omitted
- To avoid a Cartesian product, always include a valid join condition
  - in either the FROM clause or the WHERE clause

12

## Joining Three or More Tables



13

## Joining Three or More Tables: ANSI/SQL92

- Use additional join expressions in the WHERE clause
- Each join always involves two tables and common field(s)

SQL92

```
SELECT title, descr, ln
FROM writer w, article a, type t
WHERE w.writerid = a.writerid
AND a.type = t.type;
```

14

## Joining Three or More Tables: SQL99

- Use additional INNER JOIN... ON expressions
- Each INNER JOIN expression always involves two tables and common field(s)

SQL99

```
SELECT title, descr, ln
FROM writer w
      INNER JOIN article a
            ON (w.writerid = a.writerid)
      INNER JOIN type t
            ON (a.type = t.type)
ORDER BY descr;
```

15

## Practice Time

- Using the ANSI/SQL92 join style, show each instructor's full (concatenated) name, course description, and the start date/time and location for each section they're teaching. Only show sections that meet in the M building.
- Modify previous statement to use ANSI/SQL99 join style

16

## Warm Up

- ENROLLMENT is a **junction table** that breaks the Many-to-Many relationship between STUDENT and SECTION into two separate One-to-Many relationships

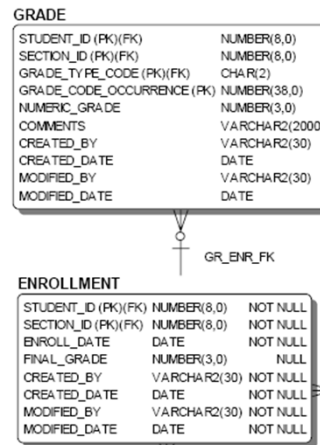
```
SELECT * FROM enrollment;
```

STUDENT_ID	SECTION_ID	ENROLL_DATE	FINAL_GRADE	C
220	280	101 21-FEB-07	(null) DSCH	
221	281	99 21-FEB-07	(null) DSCH	
222	281	101 21-FEB-07	(null) DSCH	
223	282	99 21-FEB-07	(null) DSCH	
224	282	101 21-FEB-07	(null) DSCH	
225	283	99 21-FEB-07	(null) DSCH	
226	283	101 21-FEB-07	(null) DSCH	

17

## Multicolumn Join

- ENROLLMENT and GRADE have a 1:Many relationship
- Composite primary key**
  - A primary key consisting of a combination of two (or more) fields
  - ENROLLMENT table has a **composite primary key** of STUDENT\_ID and SECTION\_ID together
- Composite foreign key**
  - A foreign key consisting of a combination of two (or more) fields
  - GRADE table has STUDENT\_ID and SECTION\_ID as a **composite foreign key** so each GRADE record can be matched to an ENROLLMENT record



18

## Multicolumn Join ANSI/SQL92

- Show student 283's last name, courses and grades

**SQL92**

```
SELECT stu.last_name, cou.description,
       gr.numeric_grade, gr.grade_type_code
FROM student stu, enrollment enr,
     section sec, course cou, grade gr
WHERE stu.student_id = enr.student_id AND
       enr.student_id = gr.student_id AND
       enr.section_id = gr.section_id AND
       enr.section_id = sec.section_id AND
       sec.course_no = cou.course_no AND
       stu.student_id = 283;
```

**composite keys**

Query Result x

All Rows Fetched: 18 in 0.032 seconds

LAST_NAME	DESCRIPTION	NUMERIC_GRADE	GRADE_TYPE_CODE
1 Perkins	Intro to the Internet	85	FI
2 Perkins	Intro to the Internet	84	HM
3 Perkins	Intro to the Internet	84	HM

## Multicolumn Join ANSI/SQL99

- Show student 283's last name, courses, grades

**SQL99**

```
SELECT stu.last_name, cou.description,
       gr.numeric_grade, gr.grade_type_code
FROM student stu
INNER JOIN enrollment enr
    ON (stu.student_id = enr.student_id)
INNER JOIN grade gr
    ON (enr.section_id = gr.section_id)
    AND (enr.student_id = gr.student_id)
INNER JOIN section sec
    ON (enr.section_id = sec.section_id)
INNER JOIN course cou
    ON (sec.course_no = cou.course_no)
WHERE stu.student_id = 283;
```

**composite keys**

Query Result x

All Rows Fetched: 18 in 0 seconds

LAST_NAME	DESCRIPTION	NUMERIC_GRADE	GRAC
1 Perkins	Intro to the Internet	85	FI

## Practice Time

---

- Provide a SQL statement that shows each instructor's last name, the section id and start date/time for sections they're teaching and, for each student enrolled in the instructor's sections, show the student's last name and numeric grade. Show only grades that are at least 80.