ANSI SQL

Joins and Their Types

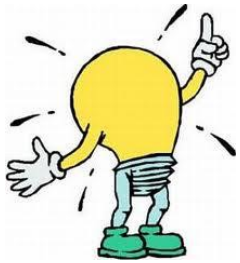
LEVEL – LEARNER

Icons Used

Hands-on Exercise

Reference

Questions



Points To Ponder

Coding Standards

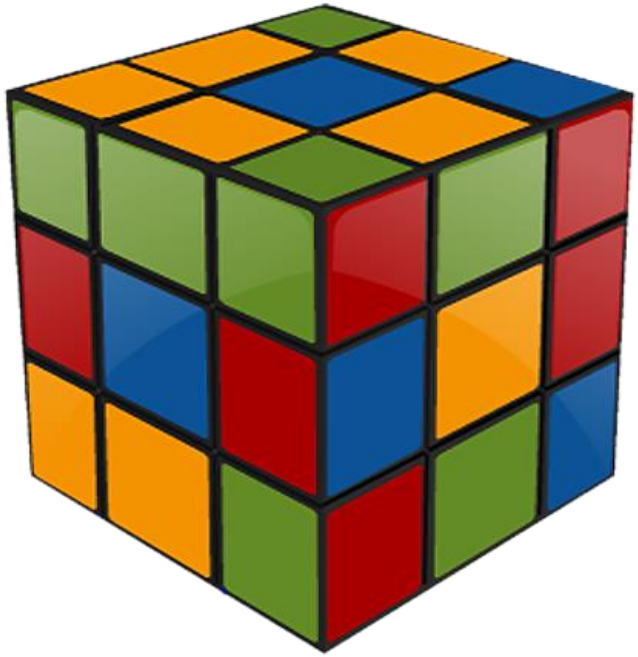
Lend A Hand

Summary

Test Your Understanding

Objectives

- At the end of this session, you will be able to:
  - Define JOIN and JOIN Style.
  - Identify Theta Style.
  - Describe ANSI Style JOIN ... ON and JOIN ... USING.
  - Define CROSS JOIN.
  - Describe INNER JOIN.
  - Recognize EQUI-JOIN.
  - Describe NATURAL JOIN.
  - Identify OUTER JOIN.
  - Define LEFT OUTER JOIN.
  - Define RIGHT OUTER JOIN.
  - Identify FULL OUTER JOIN.
  - Define SELF JOIN.



Case Study

- For a complete understanding of ANSI SQL, we are going to make use of a Product Management System (PMS) for ABC Traders.
- ABC Traders is a company which buys collectible model cars, trains, trucks, and buses, and ships them directly from manufacturers and sells them to distributors across the globe.
- In order to manage the stocking, supply, and payment transactions, the PMS mentioned above was developed.
- As per the requirement of the trading company, an inventory system is developed to collect information of products and customers and their payment processing.

Database Tables

- There are many entities involved in PMS.
- We will be dealing with the PMS given below throughout this course.

Offices
To maintain information of offices, for example, office code, address, city, and so on.

Customer
To maintain customer details, for example, customer name, address, and so on.

Employees
To maintain employee details, for example, ID, name, and so on.

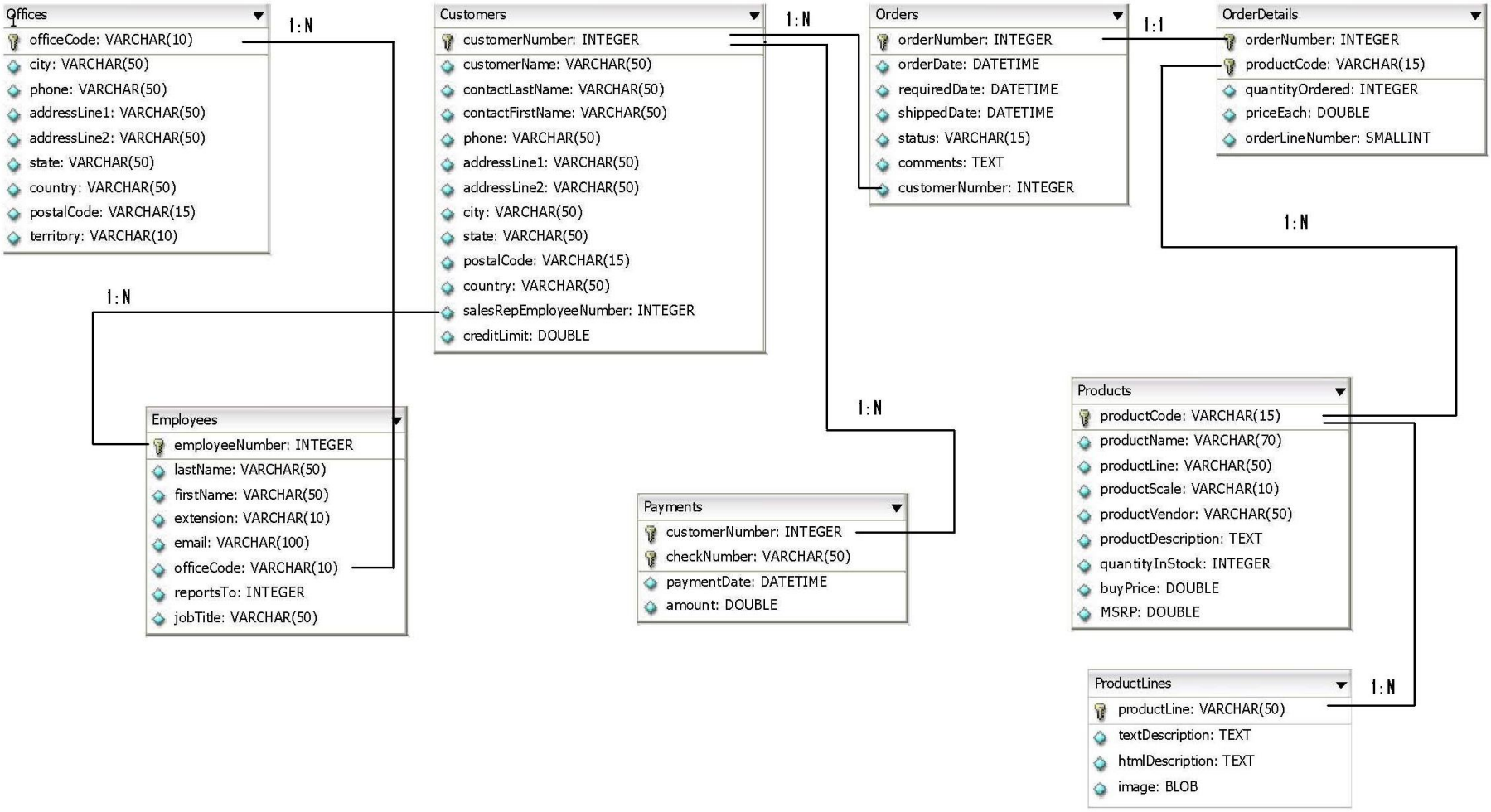
Products
To maintain information of products, for example, product ID, name, and so on.

Payments
To maintain information of payments done, for example, payment date, amount, and so on.

Orders
To maintain orders done by customers, for example, order number, date, and so on.

Order Details
To maintain orders done by customers, for example, order number, date, and so on.

Schema Diagram



Scenario

Hi! I am Tim and I am back with a few questions. Now that you have created tables with constraints and used different operators, functions, and clauses; I would want you to take care of some requirements which involve fetching data from more than one tables.

Let us learn about JOINS which will help us meet Tim's requirements.

Do You Know?

What will be the query to SELECT data from multiple tables?

- Why Join?
  - SQL JOINS are used to query data from two or more tables, based relationship between certain columns in these tables.
  - Using JOINS, you can fetch exactly the data you want from any nur of tables with just one query, using any search parameter you chose to filter the results.
  - Different vendors allow varying numbers of tables to join in a single join operation.

- For example:
  - Oracle is unlimited in the number of allowable JOINS Microsoft SQL Server allows up to 256 tables in a join operation

JOINS Style

- Which JOIN Style can be used?
  - SQL JOINS can be written using any of the two styles mentioned below:
    - Theta Style
    - ANSI Style (Preferred by Industry)

Theta Style

- Theta Style:
  - In non-ANSI standard implementation, the join operation is performed in the WHERE clause in the query.
  - This join method is known as the theta style.
  - In the FROM clause, tables are listed as if with Cartesian products.
  - The WHERE clause specifies how the join should take place.
  - This is considered to be the "old" style.
  - It is somewhat confusing to read.

Theta Style (Contd.)

- Syntax:

```
SELECT <ColumnName1>,<ColumnName2>,<ColumnNameN>
FROM<TableName1>,<TableName2>
WHERE<TableName1>.<ColumnName1>=<TableName2>.<ColumnName2>
AND<Condition>
ORDER BY<ColumnName1>,<ColumnName2>,<ColumnNameN>
```

- Where:
  - ColumnName1 in TableName1 is usually that table's Primary Key
  - ColumnName2 in TableName2 is a Foreign Key in that table
  - ColumnName1 and ColumnName2 must have the same data type and for certain data types, the same size

ANSI Style

- In the ANSI SQL-92 standards, joins are performed by incorporating JOIN clause in the query. This join method is known as ANSI style.

- Syntax:

```
SELECT <ColumnName1>,<ColumnName2>,<ColumnNameN>
FROM<TableName1><JOIN TYPE><TableName2>
ON <TableName1>.<ColumnName1>=<TableName2>.<ColumnName2>
WHERE<Condition>
ORDER BY<ColumnName1>,<ColumnName2>,<ColumnNameN>
```

- Joins in the ANSI style are more readable than those in Theta style, since the query itself clearly indicates, which table is on the left in a LEFT JOIN and which table is on the right in a RIGHT JOIN, which we will see soon.
- There are two variations of ANSI Style as JOIN ... ON (preferred by industry) and JOIN … USING which will be covered in the next example.

JOIN ... ON

- ANSI Style: JOIN ... ON
  - With JOIN ... ON, one separates the join terms from the filtering terms.

- Example:

```
SELECT o.city, o.country, e.jobtitle
FROM offices o INNER JOIN employees e
ON (o.officecode=e.officecode)
WHERE o.country LIKE '%USA%';
```

  - It is quite clear now what belongs to what.

Note: The parenthesis is not strictly required in the ON clause.

JOIN ... USING

- ANSI Style: JOIN ... USING
  - It is the special case where we join tables on columns of the same name. We can make a shortcut and use USING.

- Rewriting the previous example:

```
SELECT o.city, o.country, e.jobtitle
FROM offices o INNER JOIN employees e
USING (officecode)
WHERE o.country LIKE '%USA%';
```

  - This time the parenthesis is required.

Note: This ANSI syntax is supported only by: MySQL, Oracle, and DB2 databases.

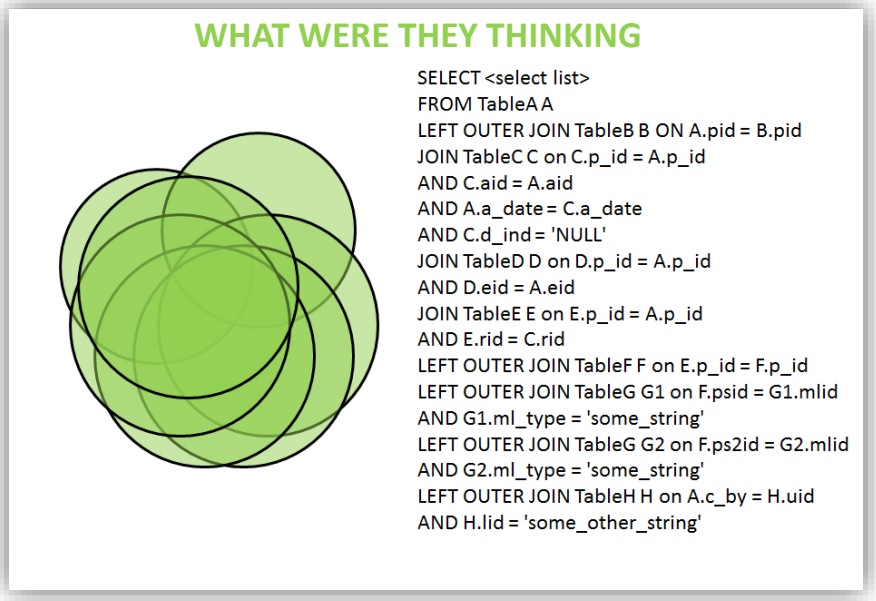
Scenario

Ok! I would want you to stick to JOIN ..ON syntax for all my requirements!

Let us use ANSI JOIN …ON syntax in all the examples.

JOIN Types

- JOIN Types:
  - We will be looking at the following types of JOINS which are supported by almost all database vendors in market
    - CROSS JOIN
    - INNER JOIN
      - EQUI-JOIN
      - NATURAL JOIN
    - OUTER JOIN
      - LEFT OUTER JOIN
      - RIGHT OUTER JOIN
      - FULL OUTER JOIN
    - SELF JOIN



ANSI standard SQL specifies four types of JOINS: INNER, OUTER, LEFT, and RIGHT.

Scenario

I want to display all combinations of country and job titles. Use: offices and employees tables.

Let us use CROSS JOIN to meet Tim's requirement.

CROSS JOIN

- CROSS JOIN returns the Cartesian product of rows from tables in the join.
- In other words, it will produce rows which combine each row from the first table with each row of the second.
- CROSS JOIN can be used in situations where the various combinations of two or more table records are needed.
- The cross join does not apply any predicate to filter records from the joined table.
- Programmers can further filter the results of a cross join by using a WHERE clause.
- ANSI Style:
  - SELECT o.country, e.jobtitle
  - FROM offices o CROSS JOIN employees e;

Rule:

If two tables t1 and t2 having columns p and r, rows n and m, respectively, are exhibiting CROSS JOIN, then the result of query will produce p+r columns and n*m tuples.

INNER JOIN

- An inner join is the most common join operation used in applications and can be regarded as the default join-type.
- Inner join creates a new result table by combining column values of two tables (A and B) based upon the join-predicate.
- The query compares each row of A with each row of B to find all pairs of rows which satisfy the join-predicate.
- When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.
- The result of the join can be defined as the outcome of first taking the Cartesian product (or Cross join) of all records in the tables (combining every record in table A with every record in table B)—then return all records which satisfy the join predicate.

Scenario

I want to display all rows from selected columns in offices and employees tables where ever there is a match between office codes. Help me do that!

Let us use INNER JOIN to meet Tim's requirement.

INNER JOIN (Contd.)

- ANSI Style:

```
SELECT o.city, o.country, e.jobtitle
FROM offices o INNER JOIN employees e
ON o.officecode=e.officecode;
```

  - The query given in the example above will join the offices and employees tables using the officecode column of both tables.
  - The query returns all rows from the offices and employees tables where there is a matching officecode value in both the tables.
  - Where the officecode does not match, no result row is generated.

INNER JOIN (Contd.)

- Programmers should take special care when joining tables on columns that can contain NULL values, since NULL will never match any other value (not even NULL itself), unless the join condition explicitly uses the IS NULL or IS NOT NULL predicates.
- INNER JOIN can be classified further into the following types. These will be discussed in the ensuing slides.
  - EQUI-JOIN
  - NATURAL JOIN

EQUI-JOIN

- An EQUI-JOIN is a specific type of comparator-based join, that uses only equality (=) comparisons in the join-predicate.
- Using other comparison operators (such as <,>,<=,>=) disqualifies a join as an EQUI-JOIN. The query shown before has already provided an example of an EQUI-JOIN.

- ANSI Style:

```
SELECT o.city, o.country, e.jobtitle
FROM offices o INNER JOIN employees e
ON o.officecode=e.officecode;
```


NATURAL JOIN

- A NATURAL JOIN is a type of EQUI-JOIN where the join predicate arises implicitly by comparing all columns in both tables that have the same column-names in the joined tables.
- The resulting joined table contains only one column for each pair of similarly named columns.
- Most experts agree that NATURAL JOINS are dangerous and therefore strongly discourage their use.
- The danger comes from inadvertently adding a new column, named the same as another column in the other table.
- An existing NATURAL JOIN might then "naturally" use the new column for comparisons, making comparisons or matches using different criteria (from different columns) than before.
- Thus an existing query could produce different results, even though the data in the tables have not been changed, but only augmented.

Scenario

I want to display a table after implicitly joining orders and order details in such a way that it contains only one order number column. Help me to do that!

Let us use NATURAL JOIN to meet Tim's requirement.

NATURAL JOIN: ANSI Style

- ANSI Style:

```
SELECT * FROM orders NATURAL JOIN orderdetails;
```

  - As with the explicit USING clause, only one ordernumber column occurs in the joined table, without qualifier table name.

Rule:
In case of USING clause and NATURAL JOIN, only one copy of join key column occurs as the result, with no qualifier.

OUTER JOIN

- An OUTER JOIN does not require each record in the two joined tables to have a matching record.
- The joined table retains each record—even if no other matching record exists.
- OUTER JOINS divides further into the following, depending on the table's rows that are retained:
  - LEFT OUTER JOIN or LEFT JOIN
  - RIGHT OUTER JOIN or RIGHT JOIN
  - FULL OUTER JOIN
    - (In this case left and right refer to the two sides of the JOIN keyword.)
- No implicit join-notation for OUTER JOINS exist in standard SQL (which implies you must use either LEFT /RIGHT/FULL, just OUTER JOIN will not work).

Scenario

I want to display selected columns from customers and payments tables in such a way that all rows from customers table are displayed irrespective of whether there is a match in payments or not.

Let us use LEFT JOIN to meet Tim's requirement.

LEFT OUTER JOIN

- The result of a LEFT OUTER JOIN (or simply left join) for table A and B always contain all records of the "left" table (A), even if the join-condition does not find any matching record in the "right" table (B).
- This means that if the ON clause matches 0 (zero) records in B (for a given record in A), the join will still return a row in the result (for that record)—but with NULL in each column from B.
- A LEFT OUTER JOIN return all the values from an inner join and all values in the left table that do not match with the right table.

- ANSI Style:

```
SELECT c.city, p.checknumber, p. amount
FROM customers c LEFT OUTER JOIN payments p
ON c.customernumber=p.customernumber;
```


Scenario

I want to display selected columns from customers and payments tables in such a way that all rows from payments table are displayed irrespective of whether there is a match in customers or not.

Let us use RIGHT JOIN to meet Tim's requirement.

RIGHT OUTER JOIN

- A RIGHT OUTER JOIN (or right join) closely resembles a LEFT OUTER JOIN, except with the treatment of the tables reversed.
- Every row from the "right" table (B) will appear in the joined table at least once.
- If no matching row from the "left" table (A) exists, NULL will appear in columns from A for those records that have no match in B.
- A RIGHT OUTER JOIN returns all the values from the right table and matched values from the left table (NULL in case of no matching join predicate).
- ANSI Style:

```
SELECT c.city, p.checknumber, p. amount
FROM customers c RIGHT OUTER JOIN payments p
ON c.customernumber=p.customernumber;
```


Scenario

Combine the effect of applying both LEFT and RIGHT OUTER JOINS on customers and payments tables.

Let us use FULL OUTER JOIN to meet Tim's requirement.

FULL OUTER JOIN

- Conceptually, a FULL OUTER JOIN combines the effect of applying both LEFT and RIGHT OUTER JOINS.
- Where records in the FULL OUTER JOINed tables do not match, the result set will have NULL values for every column of the table that lacks a matching row
- For those records that do match, a single row will be produced in the result set (containing fields populated from both tables)
- Supported: Microsoft SQL Server, DB2, Oracle 10g, 11g
- Not Supported: MySQL, Sybase
- ANSI Style:

```
SELECT c.city, p.checknumber, p. amount
FROM customers c FULL OUTER JOIN payments p
ON c.customernumber=p.customernumber;
```

 |

FULL OUTER JOIN (Contd.)

- Some database systems do not support the FULL OUTER JOIN functionality directly, but they can emulate it through the use of an INNER JOIN. UNION ALL selects the "single table rows" from left and right tables respectively.

- Example:

```
SELECT c.city, p.checknumber, p. amount FROM customers c
LEFT JOIN payments p ON c.customernumber=p.customernumber
UNION
SELECT  c.city, p.checknumber, p. amount FROM customers c
RIGHT JOIN payments p ON c.customernumber=p.customernumber;
```


SELF JOIN

- A SELF JOIN is joining a table to itself.
- Use a SELF JOIN when you want to create a result set that joins records in a table with other records in the same table.
- To list a table two times in the same query, you must provide a table alias for at least one of instance of the table name.
- This table alias helps the query processor to determine whether columns should present data from the right or left version of the table.

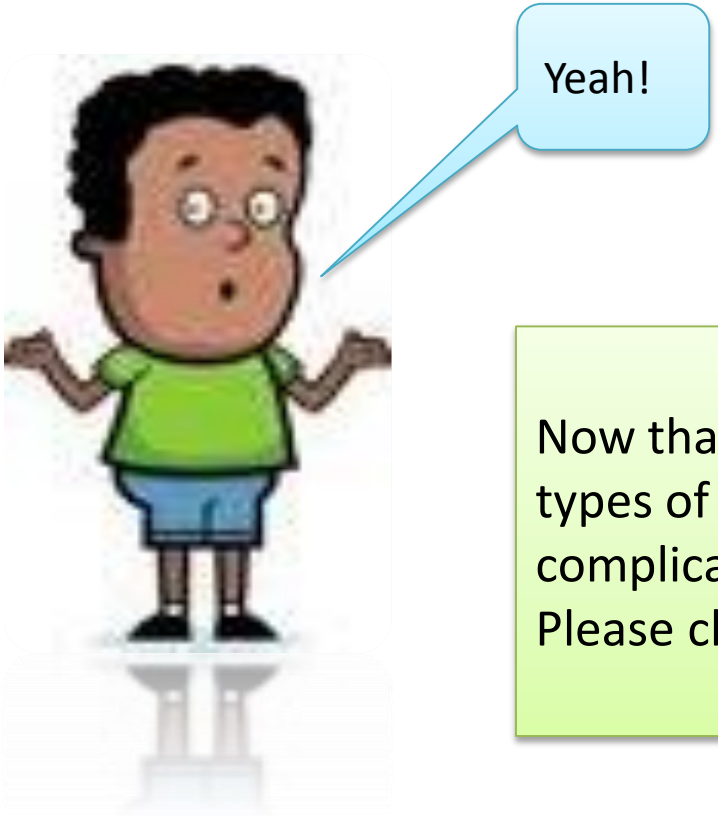
- ANSI Style:

```
SELECT a.lastname, a.reportsto, b.firstname, b.reportsto,
b.jobtitle
FROM employees a INNER JOIN employees b
ON a.jobtitle =b.jobtitle;
```


Any Questions?

 |

Hands-on



Now that we are well versed with JOIN and various types of JOINS, let's help Tim to find answers to his complicated queries for online feedback system.
Please check the hands-on document for more details.

Activity

- Now that we are well versed with the commands, let us test our understanding using a short case study.

Course Management System (CMS) Cognizant Academy

Outcome Assured…

- Case Study Scenario:
  - This case study is to develop a Course Management System (CMS) for Cognizant Academy. The following are the two use cases for which the database needs to be designed.
    - Add Course
      - To add the course details into the Course Management System.
    - Retrieve Course
      - Retrieve the courses stored in the system and display it.
  - The courses to be added will have course code, course name, number of participants, course description, course duration, course start date, and course type.

myAcademy What have you learnt today?

Lend a Hand

- Requirement #1: Write a query to fetch student ID, first name, last name, and course code for students who have enrolled for course having course_code as 167. Student_Info and student_courses to be queried.
- Requirement #2: Write a query to display the discount offered on the courses along with course descriptions.
- Pre-requisite for Requirement #3: Create a student record (student_info_ <employee id>).
- Note: Do not create any other detail regarding this student in any other table.
- Requirement #3: Write a query to fetch first names of the students along with the course codes of the courses they have enrolled in.
- Note: Even if the course_code does not exist for a student, the record needs to be fetched.
- Get the same output as per the requirement above using:
  - Left Join
  - Right Join

Lend a Hand: Solution

- Solution 1:

```
SELECT s.student_id, s.first_name, s.last_name, s.address, c.course_code
FROM  student_info INNER JOIN student_course
ON s.student_id = c.student_id WHERE c.course_code = 167;
```

- Solution 2:

```
SELECT cf.discount, ci.course_description
FROM course_fees cf INNER JOIN course_info ci
ON cf.course_code=ci.course_code;
```

- Solution 3(a):

```
SELECT si.first_name, sc.course_code
FROM student_info si LEFT JOIN student_courses sc
ON si.student_id = sc.student_id;
```

- Solution 3(b):

```
SELECT first_name, course_code
FROM student_courses  sc RIGHT JOIN student_info  si
ON sc.student_id = si.student_id;
```

 |

Test Your Understanding

What is the use of JOIN and JOIN Style?

What is the use of Theta Style?

What is the use of ANSI Style JOIN ... ON and JOIN ... USING?

What is CROSS JOIN?

What is INNER JOIN?

What is EQUI-JOIN?

What is OUTER JOIN?

Test Your Understanding

What is NATURAL JOIN?

What is LEFT OUTER JOIN?

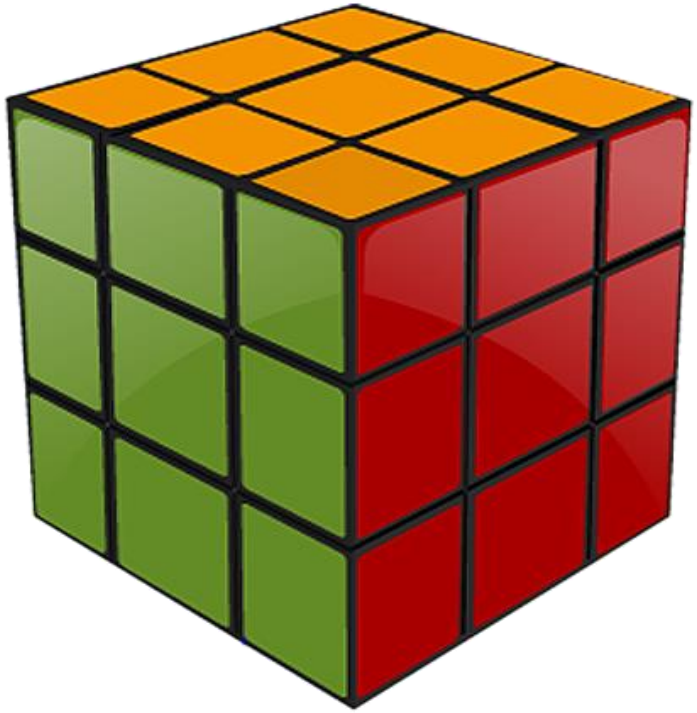
What is RIGHT OUTER JOIN?

What is FULL OUTER JOIN?

What is SELF JOIN?

Summary

- The key points covered in this chapter are:
  - SQL JOINS are used to query data from two or more tables, based on a relationship between certain columns in these tables.
  - JOINS are of two types:
    - Theta Style and ANSI style
  - CROSS JOIN returns the Cartesian product of rows from tables in the join
  - Inner join creates a new result table by combining column values of two tables (A and B) based upon the join-predicate
  - The details of Outer JOIN, CROSS JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN, and SELF JOIN.



Source

- Website:
  - http://en.wikipedia.org/wiki/Join_(SQL)

- Books:
  - O'Reilly SQL In NutShell Page No: 37, 144, 145, 146
  - Oracle for Professionals - Covers Oracle 9i, 10g and 11g W CD By Sharanam Shah, Vaishali Shah Page no:451

Change Log

Version Number	Changes made			
V1.0	Initial Version			
V1.1	Slide No.	Changed By	Effective Date	Changes Effected
	1-46	Learning Content Team CI Team CATP Technical Team	17-05-2013	Base-lining content

 |

Understanding ANSI SQL

You have successfully completed this session on Joins and Their Types