1. Relational Database Development

152-156

Structured Query Language (SQL)

Multi-Table Queries

| Notes | Activity |
| --- | --- |
| Text References   * + [Joining Tables](#Link) Pages 136 – 145   + [Qualifying Field Names](#Qualify) Pages 136 – 145   + [Using Aliases](#Alias) Page 146   + [Joins](#Inner) Pages 160 – 163   + [Linking Multiple Tables](#Multiple) Pages 150 – 152   + [Outer Joins](#Outer) Pages 161 – 163   + [Self-Joins](#Self) Pages 147 – 150   + [Subqueries](#Subquery) Pages 139 – 142   + [IN Operator](#In) Pages 140 – 141   + [Nested Subqueries](#Nested) Pages 142 – 145   + [Union](#Union) Pages 152 – 154   + [Intersect](#Intersect) Page 155   + [Minus](#Minus) Page 156   + [Any / All](#Any) Pages 157 - 160   + [Sample Query Solutions](#Solutions) |  |
| 1. Defining Links in Queries  (*Joining* Tables)    * The first step to including fields from multiple tables, is to list all the table names needed in the **From** clause of the SQL command (separate table names with commas)    * Then, in order to include fields from multiple tables in one query, those tables must be related (linked).    * The SQL command must inform the database how the tables are linked.      + This is normally done using a **Join** command      + This can also be done in the **Where** clause |  |
| 1. **Qualifying Field Names**    * Often the linking fields have the same name in both tables. This can cause problems, because SQL can’t figure which field you’re referring to : **Where** StudentID = StudentID    * To resolve this, you must *qualify* a field name if it appears in more than one table. To qualify a field name, precede it with its table name, separating them with a period (just like in most programming languages):  **Where** tblStudent.StudentID=tblEnroll.StudentID | Demonstrated in following examples  Activities colored red apply to the Employees (Trips) database.  Activities colored green apply to the WPR donations database. |
| 1. **Using Aliases to Simplify Qualification**    * Long table names in multi-table queries can make the query hard to read.    * Like fields, tables can be assigned an alias. As with fields, simply follow the table name with a space and then the alias. Do not include the keyword **As**    * Then, use the alias when qualifying field names:  **Where** st.StudentID = en.StudentID |  |
| 1. **Linking Tables Using Where**    * One way to link tables is to use the where clause    * For reference only. Unless I specify otherwise, use Join   Select FName, LName, Section  From tblStudent s, tblEnroll e  Where s.StudentID = e.StudentID   * + The tables are linked in the Where clause where the ID numbers in the tables match   + The order of the tables doesn’t seem to matter in either the From clause or the Where clause | Add Volker Gaul as an employee (plant: 1)  Display the employee name, trip date and destination for each trip, order by lastname, firstname  Note Volker Gaul doesn’t show up. |
| 1. **Linking Tables Using Join**    * The primary tool for linking tables is the Join command.   Select FName, LName, Section  From tblStudent s  Inner Join tblEnroll e  On s.StudentID = e.StudentID   * + The Join phrase tells SQL to link the student table  (alias “s”) to the enrollment table (alias “e”)     - The order of the tables doesn’t seem to make much difference. SQL determines which table is the parent and which is the child.     - For clarity sake, it’s usually better to list the parent table first.     - When we start using other kinds of joins, the order will be important and the parent is usually listed first.     - The keyword, Inner, is actually optional, but I always include it for clarity.     - Only students who have enrolled (have a record in the tblEnroll) will appear in the results of the query above.   + The On clause replaces the Where clause (from version above)     - Note you have to tell SQL what two fields are used to link the tables (On clause)     - Again, the order of the fields in the On clause doesn’t matter.   + I prefer this method because it separates the linking specification from the rest of the Where clause that is usually used to specify row selection criteria. | Repeat the query above using Join  Add Adam Aadond as a donor  Order by lastName, firstName.  Note: Aamond doesn’t show up. |
| * + Joining with Using     - If the field names in the two tables are identical, you can simplify the linking process a little with the Using clause   Select FName, LName, Section  From tblStudent  Join tblEnroll   Using (studentID) |  |
| * + - Note: Using only works when the primary key and foreign key have the same names (usually)     - ( ) required     - Using also eliminates the need to qualify fields in the field list or where clause and therefore the aliases       * Linking fields are rarely displayed anyway (except in book exercises) |  |
| * + - Note: Not all versions of SQL include a Using clause so be sure you are comfortable with the On clause       * Access for instance |  |
| 1. **Linking More than Two Tables Using Join**    * When using Join to connect multiple tables, you create a *chain* of Join clauses   **Select** Concat(FName, ' ', Lname) As Name,  CourseName  **From** tblStudent  **Inner Join** tblEnroll Using (StdID)  **Inner Join** tblCourse Using (CrsNumber)   * + I suggest you use the ERD as a guide to help you create the linking statements.   + Note: when creating data table queries in Visual Studio (Access database), VS seems to want the initial Join clause to be surround by parenthesis. | List last and first name (concatenated), trip date, expenses, plantname, and foreman for all trips to Chicago  Add the station call letters to the previous query. |
| 1. **Linking More than Two Tables Using Where**    * For every related pair of tables in the query, you must provide the link in the **Where** clause. The **And** keyword ensures all links are created.   **Select** Concat(FName, ' ', Lname) As Name,  CourseName  **From** STUDENT s, ENROLL e, COURSE c  **Where** s.StdID = e.StdID  **And** e.CrsNumber=c.CrsNumber | Repeat using Where  Note how the Where clause does both linking and filtering  Add parenthesis around the first join clause |
| 1. **Outer Joins**    * Some queries that join tables require that all the records of one table (usually the *parent* table) be displayed even if there are no matching (*child*) records.    * Examples      + List the total sales for all sales people this month. List sales people even if they haven't sold anything yet.      + List all students' scores on an assignment, even those that haven't submitted it yet.    * For the rows where the parent table doesn't have matching child records, the data for the (missing) child record will appear as NULL    * Outer joins come in three flavors      + Left Join (note: the word *outer* isn't used)        - Parent table is listed first (on the left) in the join      + Right Join        - Parent table is listed second (on the right) in the join      + Full Join        - Not supported by MySQL        - Non-matching records from BOTH tables are included        - Why would there be child records without a parent? That's the point—find the child records without a parent (orphan records)    * The parent table doesn't have to be listed as the primary table (the child can be), but normally, Left or Right designates where the parent is (which table should include all records).    * If you want to add a criteria to an outer join, you MUST use an AND clause after the ON clause, not WHERE in order to show all records from the outer join table.      + Using cannot be combined with And | List the dates that each employee travelled  Display number of donations made by each donor.  First use a Join.  Note Volker Gaul missing and Adam Aamond  Use Outer Join  Note Volker Gaul value is Null  Adam Aamond is 0  Limit the results to only include trips taken to Chicago (continue to show all employees) (use and, not where)  Donations in 2009 only. |
| * + Examples:  Select CustLastName, CustFirstName, Sum(OrderAmt) As TotalSales   From tblCustomer  --Include all the records from tblCustomer  Left Join tblOrders Using (CustID)  Group By CustLastName, CustFirstName  Order By TotalSales Desc  Select CustLastName, CustFirstName, Sum(OrderAmt) As TotalSales  From tblOrders o  --Include all the records from tblCustomer  --Can’t use Using because there’s an AND clause  Right Join tblCustomer c On o.CustID=c.CustID  And OrderDate>’04-01-2014’  Group By CustLastName, CustFirstName  Order By TotalSales Desc | |
| 1. **Joining a Table to Itself**    * Circumstances arise when you want to join a table to itself.    * Examples:      + Parts that are made up of other parts. “List all the parts that make up part XYZ.”      + People who supervise people. “List all of Bill Smith’s subordinates.”    * Linking a table to itself is also a good way to look for duplicate data in tables that use *manufactured* or *autonumber* fields. For example, if a customer table is keyed on a customer number, it is very easy to mistakenly enter the same customer twice with different keys.    * To link a table to itself,      + List the table in the **From** clause **twice** and give each a different alias.      + List the fields you want twice, each linked to a different alias.      + Link the tables in the following manner:  **Inner** Join table2 t2 On t1.key < t2.key or Where t1.key < t2.key         - If you use the = as you normally do for linking, you’ll get 4 matches for every pair, 2 for records matching themselves, 1 where the first record is listed first, then the second, and 1 where the second record is listed first, then the first.        - Using less than ensures each pair of matching records is only listed once. | Select the employeeId, TripDate and Destination where the TripDate and Destination are the same.  List the first name and last name of each donor pair that have the same first name. |
| 1. **Subqueries**    * A query can be based on the results of an embedded query. The embedded query (SQL command) is analyzed first and then the primary query uses those results to obtain its answer.    * The embedded query is contained in the primary query’s **Where** clause (surrounded by parenthesis).      + The embedded query is evaluated first, creating a temporary table of records.      + The main query then processes, using the temporary query results as its row criteria.    * Subqueries can only return the values from one column.    * Queries can be *nested* to as many levels as necessary (query inside a query, inside a query…) |  |
| * + One common use of subqueries is to determine which records are related to a statistic (MIN, MAX, etc).  ”Determine which customer has the highest credit limit.”  ”List the students who scored lower than the average on the test.”  **Select** custfirst, custlast **From** tblCustomer   **Where** creditlimit =  (**Select** **Max**(creditlimit) **From** tblCustomer)  **Select** studentfirst, studentlast **From** tblTest  **Where** score <  (**Select** **Avg**(score) **From** tblTest) | Show the employee who has the highest salary.  Show the donor name who made the largest donation (2 nested queries) |
| * + Another common use of subqueries is to link data from two tables.  ”Show where the Gateway computer is located.”  The information we need (location) is in a different table than the information we’re given (Gateway). To solve this problem, we’ll first determine the CompID for the Gateway, then use that to determine its location.   **Select** Location **From** PC  **Where** CompID =  (**Select** CompID **From** COMPUTER  **Where** MfgName='Gateway')  Alternate solution using a join:  **Select** Location  **From** PC **Inner** **Join** COMPUTER  **Using** (CompID)  **Where** MfgName='Gateway'  Most subqueries used to link two tables can be done using a join (and usually a little more clearly). | Show what state Volker Gaul works in (using a subquery)  Show the trips taken by Yacob Meiser (using a subquery)  Show the trips to Chicago whose expenses are greater than the average expenses for all trips  Show the station information for the station Volker Gaul listens to using a subquery.  Show the donations made by Volker Gaul using a subquery. |
| * + Though linking can be done without subqueries, subqueries allow you to break complex queries into parts, often simplifying or at least organizing the parts of the query. | Repeat using a Join  Note the Max & Avg query examples cannot be done with a Join |
| 1. **Using the IN Operator**    * The subquery example above works because there is only one Gateway computer installed. If we wanted to know the locations of the two OmniTech computers, this query would fail because there are two CompIDs with MfgName Gateway. SQL wouldn’t know which to use so it returns nothing.    * The In operator is the solution to this problem. Remember the In operator selects all records where the field value is any one of many values.  **Where** state **In** ('MI', 'WI', 'MN')    * By replacing the list with a subquery, we use a list generated from another table.   **Select** lname, fname **From** Customer **Where** state **In**  (**Select** state **From** assignment  **Where** SlsRep='11111');  This example shows all the customers who reside in the states assigned to sales rep 11111. The subquery selects the states for this sales rep; the main query shows the customers who live in those states. | Show the employees who work in Michigan  Show the donors (and station ID) who listen to FM stations |
| * + Note subqueries used in an **IN** clause must still only return one field, though that field could contain multiple values.   + The **IN** clause can often simplify queries where the criteria includes (fields from) tables that are not needed in the field list. | Repeat using a Join |
| 1. **Nested Subqueries**    * As mentioned above, subqueries can be nested (query inside of a query inside of a query, etc).    * The lowest level queries are always processed first.   **Select** lname, fname **From** Customer **Where** state **In**  (**Select** state **From** assignment  **Where** SlsRep **In**  (**Select** SlsRep FROM Staff  **Where** lname = 'Gaul') )   * + This query has the same results as the one in the previous section assuming Gaul is salesrep number 11111. Note in this case you don’t needed to know Gaul’s sales rep number to find the customers.   + Also note, I used the IN operator to find the SlsRep. Since Gaul is only going to have one number, I could have used the equal sign. The IN operator works in all cases, the equal sign only works if there’s only one matching record. | List the foremen of the employees who have traveled to Chicago using a subquery.  Note automatic Distinct  List the favorite program of each customer who listens to the Wausau station (3 levels)  Repeat using a Join  Make Distinct |
| 1. **Set Operations**    * Set operations take the results of two queries and combine them into one result (using set theory concepts).    * This is different than subqueries. Subqueries use the result of one query as the starting point for the next. Set operations combine the results of two queries.    * Each query results in a temporary result table    * Each query must list the same set of fields (or they can’t be combined) |  |
| * + **Union**     - The Union operator adds the results of the second query to the first.     - Union operators simplify queries that link tables, but also have an OR condition.     - Union queries can often be duplicated using joins and Or compound conditions, but not always.     - Can add an Order By clause to end of the union | Discuss example from GradSurvey (grads and apprentice)  Display Destination, Date and Expenses for all trips over $900 combined with all the trips to New York.  Sort by Date  Display a list of employees who have traveled to either Chicago or New York |
| * + **Intersect**      - MySQL does not support Intersect. Use a subquery.     - The Intersect operator returns only those records that appear in both temporary tables.     - Normally, the query requirements will include word “AND”.     - Duplicate using an In subquery | List employees who have traveled to both Chicago and New York  (Note: cannot be done using just a join) |
| * + **Minus**      - MySQL does not support Minus. Use a subquery.     - The Minus operator returns the records from the first query with any matching records from the second query removed.     - Normally, the query requirements will include the words “BUT NOT”     - Duplicate using a Not In subquery | List employees who have traveled to Chicago but not New York |
| * + **Any and All**     - Any and All allow you to create queries that compare a value in a record with a list of values supplied by a subquery.     - As far as I can tell, Any and All queries can alternatively be accomplished using (Select Min(value) From table) or Max(Value) subqueries appropriately     - Examples: |  |
| --Select all customers whose credit limit is greater than the largest  --credit limit of the customers from Plover  Select custfirst, custlast From tblCustomer  Where creditlimit >  (SELECT Max(creditlimit) From tblCustomer  Where City='Plover')  --Using ALL this time  Select custfirst, custlast From tblCustomer  Where creditlimit > ALL  (Select creditlimit From tblCustomer  where City='Plover')  --Select the people who are older than any person in the sophomore  --class.  Select firstName, lastName, Age From tblStudent  Where Age >  (Select Min(Age) From tblStudent  Where Class=10)  --Using ANY this time  Select firstName, lastName, Age From tblStudent  Where Age > ANY  (Select Age From tblStudent  Where Class=10) | |
| In class examples:  Display the employees who were hired after the last employee hired in Polonia.  Display the trip information for any trip taken on Feb 27, where the expenses were greater than any trip taken to Fort Worth.  If time allows redo using Min and Max |  |

|  |
| --- |
| 1. **Query Solutions** |
| * + Simple Join   Select LastName, FirstName, TripDate, Destination  From tblEmployee  Inner Join tblTrips Using (empId);  Select LastName, FirstName, TripDate, Destination  From tblEmployee e  Inner Join tblTrips t On e.empId=t.empId;  Select LastName, FirstName, TripDate, Destination  From tblEmployee e, tblTrips t   1. Where e.empId=t.empId; |
| * + Multi-Table Join (more than 2)   Select Concat(LastName, ', ', FirstName) As Employee,  TripDate, Expenses, p.Plant, Foreman  From tblPlant  Inner Join tblEmployee Using (plantId)  Inner Join tblTrips Using (empId) Where Destination=’Chicago’;  Select Concat(LastName, ', ', FirstName) As Employee,  TripDate, Expenses, p.Plant, Foreman  From tblPlant p, tblEmployee e, tblTrips t  Where p.plant=e.plant  And e.EmpID=t.EmployeeID  And Destination=’Chicago’; |
| * + Outer Join   Select lastName, firstName, tripDate  From tblEmployees e Left Join tblTrips t On e.empId=t.empId  And destination='Chicago'  Order By lastName, firstName;  (List of people who have never been to Atlanta. Note use of And and Where.) Select lastname, firstname, state, tripdate, destination  From tblplants  Inner Join tblemployees Using (plantid)  Left Join tbltrips On empid=employeeid  And destination ='Atlanta'  Where tripdate is null  Order By lastname, firstname, tripdate desc; |
| * + Self Join   Select t1.EmployeeID, t2.EmployeeID, t1.TripDate, t1.Destination  From tblTrips t1  **Inner** Join tblTrips t2 On t1.EmployeeID<t2.EmployeeID  Where t1.Destination=t2.Destination  And t1.TripDate=t2.TripDate   1. Order By t1.TripDate Desc, t1.Destination;  Select Concat(e1.LastName,', ',e1.FirstName) As Employee1, 2. Concat(e2.LastName,', ',e2.FirstName) As Employee2, 3. t1.TripDate, t1.Destination 4. From tblTrips t1 5. **Inner** Join tblTrips t2 On t1.EmployeeID<t2.EmployeeID 6. **Inner** Join tblEmployee e1 On t1.EmployeeID=e1.EmpID 7. **Inner** Join tblEmployee e2 On t2.EmployeeID=e2.EmpID 8. Where t1.Destination=t2.Destination 9. And t1.TripDate=t2.TripDate 10. Order By t1.TripDate Desc, t1.Destination; |
| * + Subqueries   Select \* From tblEmployee  Where Salary=  (Select Max(Salary) From tblEmployee);  Select State From tblPlant  Where Plant=  (Select Plant From tblEmployee  Where LastName='Gaul');  Select \* From tblTrips  Where EmployeeID=  (Select EmpID From tblEmployee  Where LastName='Meiser' And FirstName='Yacob');  Select \* From tblTrips  where destination='Chicago' And  expenses >  (Select Avg(expenses) From tbltrips);  Select \* From tblTrips t  Inner Join tblEmployee e On e.EmpID=t.EmployeeID  Where LastName='Meiser' And FirstName='Yacob'; |
| * + In   Select Concat(LastName, ', ', FirstName) As Employee, Plant  From tblEmployee  Where Plant In  (Select Plant From tblPlant  Where State='MI');  Select Concat(LastName, ', ', FirstName) As Employee, p.Plant  From tblEmployee e  Inner Join tblPlant p Using (plantId)  Where State='MI'; |
| * + Nested Queries   Select Foreman From tblPlant  Where Plant In  (Select Plant From tblEmployee  Where EmpID In  (Select EmployeeID From tblTrips  Where Destination='Chicago'  )  );  Select Distinct Foreman  From tblPlant p  Join tblEmployee e Using (plantId)  Join tblTrips t On e.EmpID=t.EmployeeID  Where Destination='Chicago'; |
| * + Union   Select Destination, TripDate, Expenses From tblTrips  Where Expenses>=900  Union  Select Destination, TripDate, Expenses From tblTrips  Where Destination='New York'  Order By TripDate, Destination;  Select Distinct e.EmpID, FirstName, LastName From tblemployees e  Inner Join tbltrips t Using (EmpID)  Where destination="Chicago"  Union  Select Distinct e.EmpID, FirstName, LastName From tblemployees e  Inner Join tbltrips t Using (EmpID)  Where destination="New York"  Order By LastName, FirstName; |
| * + Intersect   Select Distinct e.EmpID, FirstName, LastName From tblemployees e  Inner Join tbltrips t Using (EmpID)  Where destination="Chicago"  Intersect  Select Distinct e.EmpID, FirstName, LastName From tblemployees e  Inner Join tbltrips t Using (EmpID)  Where destination="New York"  Order By LastName, FirstName;  Select Distinct e.EmpID, FirstName, LastName From tblemployees e  Inner Join tbltrips t Using (empId)  Where destination="Chicago"  And e.EmpID In  (Select e.EmpID From tblemployees e  Inner Join tbltrips t Using (EmpID)  Where destination="New York")  Order By LastName, FirstName; |
| * + Minus   Select Distinct e.EmpID, FirstName, LastName From tblemployees e  Inner Join tbltrips t Using (EmpID)  Where destination="Chicago"  Minus  Select Distinct e.EmpID, FirstName, LastName From tblemployees e  Inner Join tbltrips t Using (EmpID)  Where destination="New York"  Order By LastName, FirstName;  Select Distinct e.EmpID, FirstName, LastName From tblemployees e  Inner Join tbltrips t Using (EmpID)  Where destination="Chicago"  And Not e.EmpID In  (Select e.EmpID From tblemployees e  Inner Join tbltrips t Using (EmpID)  Where destination="New York")  Order By LastName, FirstName; |
| * + All / Any   Select EmpID, LastName, FirstName, HireDate From tblEmployees  Where HireDate > All  (Select HireDate From tblEmployees  Where Plant="Polonia")  Order By HireDate;  Select \* From tblTrips  Where expenses > Any  (Select expenses From tblTrips Where destination ="Fort Worth")  And tripdate = '2010-02-27'  Order By tripdate Desc, Expenses Desc;  Select EmpID, LastName, FirstName, HireDate From tblEmployees  Where HireDate >  (Select Max(HireDate) From tblEmployees  Where Plant="Polonia")  Order By HireDate;  Select \* From tblTrips  Where expenses >  (Select Min(expenses) From tblTrips Where destination ="Fort Worth")  And tripdate = '2010-02-27'  Order By tripdate Desc, Expenses Desc; |