macromedia
COLDFUSION 5
The Fastest Way to Build and Deploy Powerful Web Applications

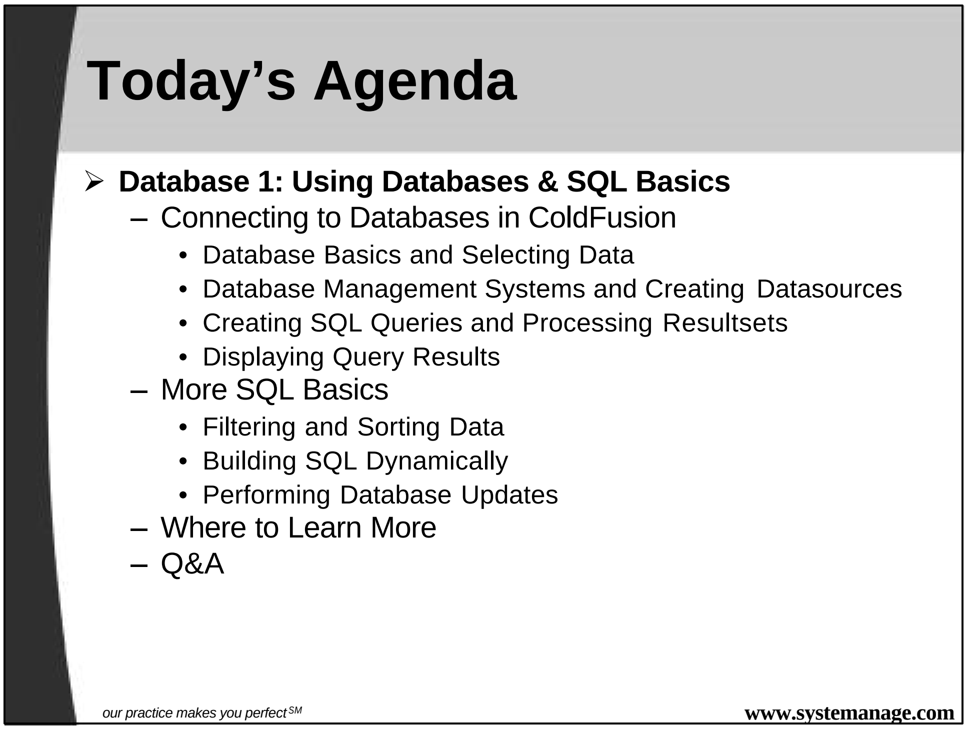
Database 1: Using Databases & SQL Basics

Charlie Arehart
Founder/CTO Systemanage
carehart@systemanage.com

SysteManage: *our practice makes you perfect* SM

Part 1 of 3

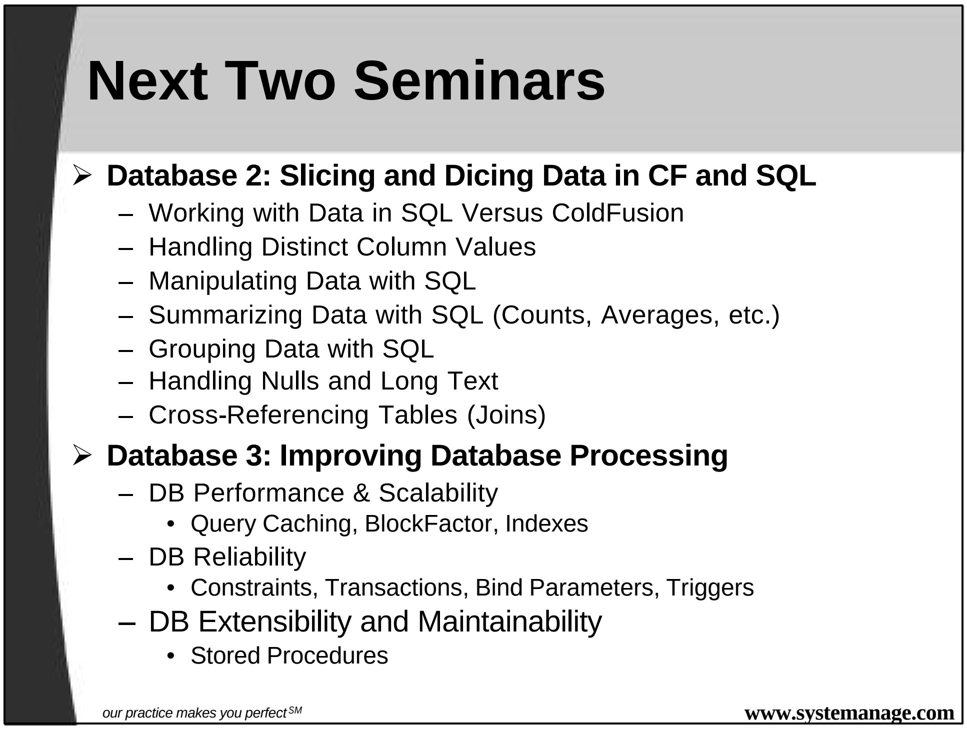
- **This seminar is part 1 of 3 being presented today**
  – First two are in conference "beginner" track
    • Database 1: Using Databases & SQL Basics
    • Database 2: Slicing and Dicing Data in CF and SQL
  – Part 3 is in "Advanced" track
    • Database 3: Improving Database Processing
- **CF experience is presumed**
  – But aspects of CF used are easy enough to pick up
- **Many topics are not really CF-specific**
  – May apply just as well to J2EE, ASP, PHP developers



Today’s Agenda

- **Database 1: Using Databases & SQL Basics**
  - Connecting to Databases in ColdFusion
    - Database Basics and Selecting Data
    - Database Management Systems and Creating Datasources
    - Creating SQL Queries and Processing Resultsets
    - Displaying Query Results
  - More SQL Basics
    - Filtering and Sorting Data
    - Building SQL Dynamically
    - Performing Database Updates
  - Where to Learn More
  - Q&A

*our practice makes you perfect*SM www.systemanage.com



Next Two Seminars

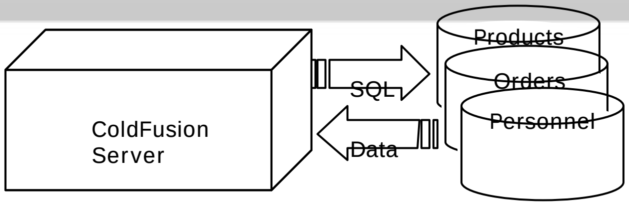
- **Database 2: Slicing and Dicing Data in CF and SQL**
  - Working with Data in SQL Versus ColdFusion
  - Handling Distinct Column Values
  - Manipulating Data with SQL
  - Summarizing Data with SQL (Counts, Averages, etc.)
  - Grouping Data with SQL
  - Handling Nulls and Long Text
  - Cross-Referencing Tables (Joins)
- **Database 3: Improving Database Processing**
  - DB Performance & Scalability
    - Query Caching, BlockFactor, Indexes
  - DB Reliability
    - Constraints, Transactions, Bind Parameters, Triggers
  - DB Extensibility and Maintainability
    - Stored Procedures

*our practice makes you perfect*SM www.systemanage.com

Logistics

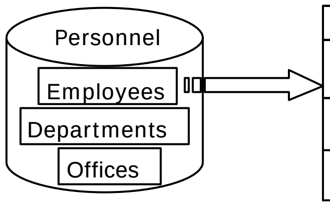
- **Database 2: Slicing and Dicing Data in CF and SQL**
  - At 10:30, in MidTown
- **Database 3: Improving Database Processing**
  - At 2:45, in Green
- **Seminars include more than just topics in brochure**
  - Indeed, had to move "joins" to Database 2 session
  - Actually, DB-2 will be really useful for even experienced developers
    - Covering many topics of SQL and CF to solve common problems often done in laborious and ineffective ways

Connecting to Databases in CF



- **Databases are the heart of most business applications**
  - Either you have one, or will create one
    - Creating databases is beyond scope of class
- ***SQL*: standard language for database access**
  - Structured Query Language (for both queries and updates) has existed for decades, now widely used
  - CF relies on using SQL for DB connection
    - Makes it very easy to create, process SQL
  - Seminars will focus on SQL and DB features of ColdFusion

Database Basics



Employees			
EmpID (Identity)	Name (Text 10)	HireDate (Date)	Salary (Currency)
1	Bob	06-04-98	$35,000
2	Cindy	12-01-00	$40,000

- ***Database*: collection of data stored in some organized fashion**
  - Composed of *tables,* structured containers holding data about a specific subject
  - Tables organized into *columns* containing particular kind of information, with an associated datatype
  - *Datatype* defines type of data column can hold
    - Examples of datatypes: text, date, currency
  - Data is stored in *rows*

Primary Keys

Employees			
EmpID (Identity)	Name (Text 10)	HireDate (Date)	Salary (Currency)
1	Bob	06-04-98	$35,000
2	Cindy	12-01-00	$40,000

- **Every row should have some column(s) to uniquely identify it, called the primary key**
  - Not required, but needed to be sure to find given record
  - Can be composed of one or multiple columns
- **Primary Key characteristics:**
  - No two rows can have the same primary key value
  - Every row must have a primary key value (no nulls)
  - The column containing primary key value cannot be updated
  - Primary key values can never be reused

Selecting Data

- **SQL's SELECT statement is most frequently used**
  - Retrieves data from one or more tables
  - At minimum, takes two clauses:
    - The data to be retrieved
    - The location to retrieve it from
  - May also specify:
    - Filtering conditions (to restrict data being retrieved)
    - Sort order (to specify how returned data is sorted)

Specifying Data to Retrieve

```
SELECT Name, HireDate, Salary
FROM Employees
```

- **Specify data to be retrieved by listing table column names as first clause of SELECT**
  - Must specify at least one column; no standard maximum allowed
  - Can specify as many as DBMS will allow
  - Can also retrieve all columns in table with `SELECT *`
    - Generally, should retrieve just the columns you need
- **Some databases require table names to be fully qualified**
  - With a prefix indicating the table owner and/or database

Renaming Columns

- **Can rename a column while selecting, using the AS keyword following column to be renamed:**
    - `SELECT Name as Empname`
- **Typically used to give names to results created with features such as aggregate functions**
    - Covered in Database 2 seminar
- **Also useful when column in database table has name that would be illegal in ColdFusion**
    - Will learn later how CF treats column names as variables
    - CF variable names cannot contain spaces, special chars
    - Some databases allow them, so AS keyword can help:
        - `SELECT [First Name] as Fname`

our practice makes you perfect SM **www.systemanage.com**

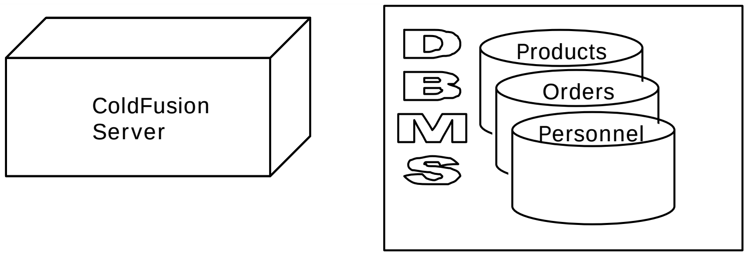
Creating Calculated Fields

```
SELECT OfficeName, Country & ‘-’ & State AS CountryState
FROM Offices
```

- **Can concatenate two or more columns together using the & operator**
    - Joins the two columns together with no space between
    - Can provide another string to be concatenated
- **Can also perform mathematical calculations on numeric columns, supporting typical operations such as +-*/ as in:**
  `SELECT Name, Salary * 1.10 as AdjSalry`
- **Will typically need to create alias to refer to calculated fields**

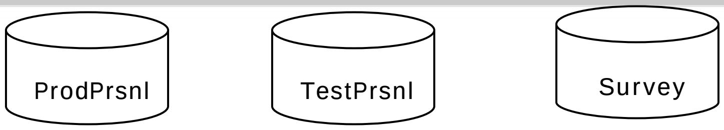
our practice makes you perfect SM **www.systemanage.com**

Database Management Systems



- ***Database Management Systems*** **organize databases into vendor-defined layout, physical file representation**
  - May run as separate server from CF, or be a simple file
- ***Database Drivers*** **provide means to communicate with DB**
- **ColdFusion hides these details from the programmer**
  - "Datasource" definition describes physical characteristics

Datasources: Logical Names



ProdPrsnl	TestPrsnl	Survey
DBMS: SQL Server	**DBMS**: SQL Server	**DBMS**: MS Access
DB Name: Personnel	**DB Name**: Personnel	**DB Name**: Surveys
Servername: prodserver	**Servername**: testserver	**Filename**: surveys.mdb
Driver: OLE-DB	**Driver**: OLE-DB	**Driver**: ODBC

- ***Datasource*****: logical name for physical DB**
  - Describes DBMS, name, physical location, database driver details for connecting to DB
    - Can choose any name, unique to CF Server
  - CF programmer needs only datasource name (DSN)
    - May need to create DBMS-specific or driver-specific SQL
    - We'll focus on very standard SQL in this series

Creating Datasources

- **Typically defined in ColdFusion Administrator**
  - Usually performed by person with admin role
  - Can also be defined in Control Panel>OBDC on Windows platforms
    - CF Administrator can edit, delete these
  - Databases requiring “native drivers” may require installation of other client libraries in support
- **Various datasource and driver characteristics can be set, to affect performance and features**
  - Default username and password can be specified
  - SQL operations can be restricted
- **See CF manuals (online and print) for details**
  - “Installing and Configuring ColdFusion Server”

Creating SQL Queries

```
<CFQUERY DATASOURCE=“ProdPrsnl”
         NAME=“GetEmployees”
         USERNAME=“#request.username#”
         PASSWORD=“#request.password#”>
   SELECT Name, HireDate, Salary
   FROM Employees
</CFQUERY>
```

- **CFQUERY tag in ColdFusion used to prepare and submit SQL to DBMS for processing**
  - Attributes can override settings in datasource definition
  - Can pass any SQL that’s acceptable to driver/DBMS
  - *DATASOURCE* attribute indicates the DSN to use
- **When CFQUERY executes a SELECT statement, it returns a result set that can be processed with CFML**
  - *NAME* attribute provides a name for that resultset

Query Result Sets

- ***Resultset* can be visualized as a table of rows and columns**
  - Stored in ColdFusion memory, after retrieval from DBMS
- **Converted to a ColdFusion query object**
  - Neither an array nor a structure, though it exhibits characteristics of both and might be thought of as an array of structures
  - Referred to by the NAME given it in the CFQUERY
  - Column names become available as variables, within a scope indicated by that NAME, as in:
    - `#GetEmployees.HireDate#`

Query ResultSet Variables

- **Query resultsets also create an associated set of variables describing the query:**
  - *RecordCount*: number of records found
  - *ColumnList*: comma-delimited list of column names
  - *ExecutionTime*: how long the query took to execute and return its results to ColdFusion, in milliseconds
- **And one variable describing each row:**
  - *CurrentRow*: number indicating the relative location of the current record within the resultset
    - This is **not** related to any internal DBMS recordid

Displaying Query Results

- **<CFOUTPUT> tag used in ColdFusion to display variables and other expressions**
  - Can be used to display query results
    - Either the first record, a particular record, or all records
- **To show the first record, use simple CFOUTPUT:**

```
– <CFOUTPUT>
  #GetEmployees.HireDate#
  </CFOUTPUT>
```

- **To show a particular record, use array notation:**

```
– <CFOUTPUT>
  #GetEmployees.HireDate[10]#
  </CFOUTPUT>
```

  - Refers to the 10th record in the resultset (again, not internal recordid, just the 10th record relative to beginning of resultset)

our practice makes you perfect SM **www.systemanage.com**

Looping Through All Records

```
<CFOUTPUT QUERY="GetEmployees">
       #Name# - #HireDate#<br>
</CFOUTPUT>
```

- **To show all records, can use QUERY attribute:**
  - Automatically loops over all records in resultset, with each iteration looking at next record
    - Note that we don't need to use queryname prefix on columns: queryname is set as default scope
    - It's still a good practice to specify it to avoid doubt
  - Be aware of need to use HTML to control appearance (perhaps <br> tag to cause newline)

our practice makes you perfect SM **www.systemanage.com**

HTML Table Formatting

```
<TABLE>
<CFOUTPUT QUERY="GetEmployees">
        <TR><TD>#Name#</TD><TD>#HireDate#</TD></TR>
</CFOUTPUT>
<TABLE>
```

- **Can also format output within HTML table**
  - Need to be careful about what is and isn't to be placed within CFOUTPUT tags
    - TABLE tags should be outside of loop
    - TR tags should be just inside beginning/end of loop
    - TD tags typically surround each column being shown

*our practice makes you perfect*SM **www.systemanage.com**

Alternating Table Row Colors

```
<TABLE>
<CFOUTPUT QUERY="GetEmployees">
    <TR <CFIF currentrow mod 2>BGCOLOR="silver"</CFIF>>
    <TD>#Name#</TD><TD>#HireDate#</TD></TR>
</CFOUTPUT>
<TABLE>
```

- **Can even alternate colors for every other table row**
  - Note that the IF test is within the <TR> tag
  - Providing a BGCOLOR="silver" attribute whenever the currentrow is odd
    - "currentrow mod 2" means divide currentrow by 2 and look at the remainder.
    - If it's not 0, then currentrow is odd

*our practice makes you perfect*SM **www.systemanage.com**

More SQL Basics

- **Examples thus far have been very simple**
  - Selecting one or more columns for all rows in table, with results returned in no defined order
- **Will conclude this seminar with a few more basic operations:**
  - *filter* data to select only desired records
  - sort results into a particular *order*
  - build SQL dynamically, at run time
  - perform not just queries but also inserts, updates, and deletes

Filtering Data

- **Can choose to select only desired records (filter the results) by way of a *WHERE* clause**
- **For instance, to find the employee with EmpID=1:**

```
SELECT Name, HireDate, Salary
FROM Employees
WHERE EmpID=1
```

  - Notice that you can filter on columns you don't SELECT
  - If datatype of column being filtered is numeric:
    - the value is specified without quotes
  - If datatype is some sort of character type:
    - the value is specified with quotes, as in:

      ```
      WHERE Name='Bob'
      ```
    - Notice that is some DBMS's, double quotes may be allowed
    - Whether dates should be quotes, and how they should be formatted, also varies by DBMS/driver
- **Can certainly filter on more than just equality matches...**

Common Filter Operators

- Common filter operators include:

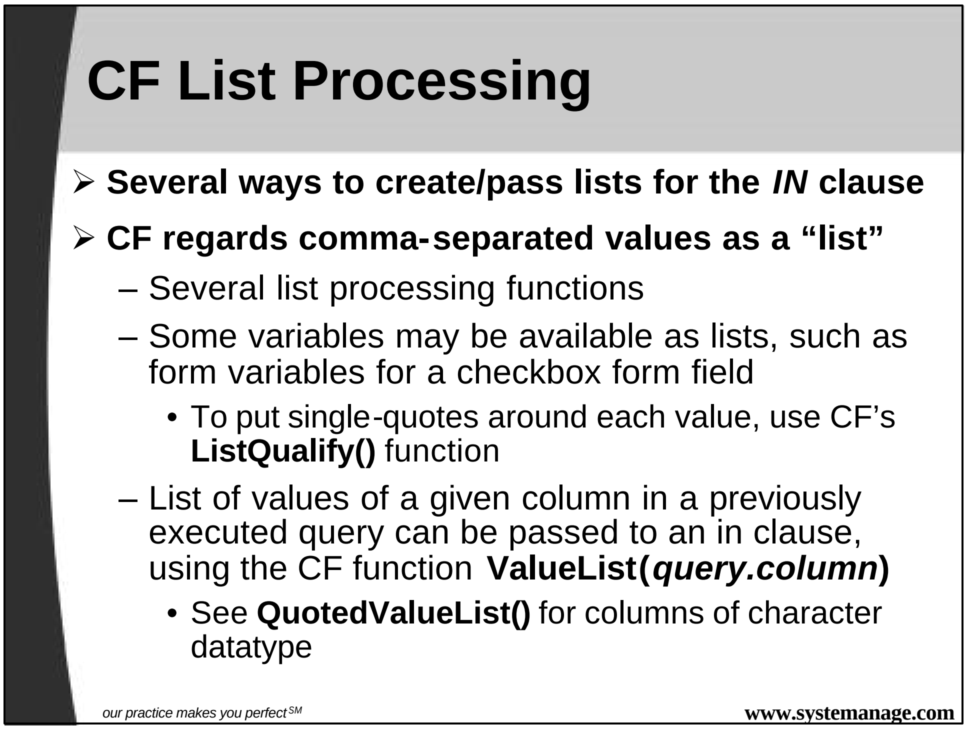
WHERE Clause Operators	
=	Equal
<>	Not equal
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
IN	One of a set of
LIKE	Matching a wildcard
BETWEEN	Between specified values
IS NULL	Is a NULL value
AND	Combine clauses
OR	Or clauses
NOT	Negate clauses

Matching on Multiple Values

- Can search for a match on multiple values using the IN clause:

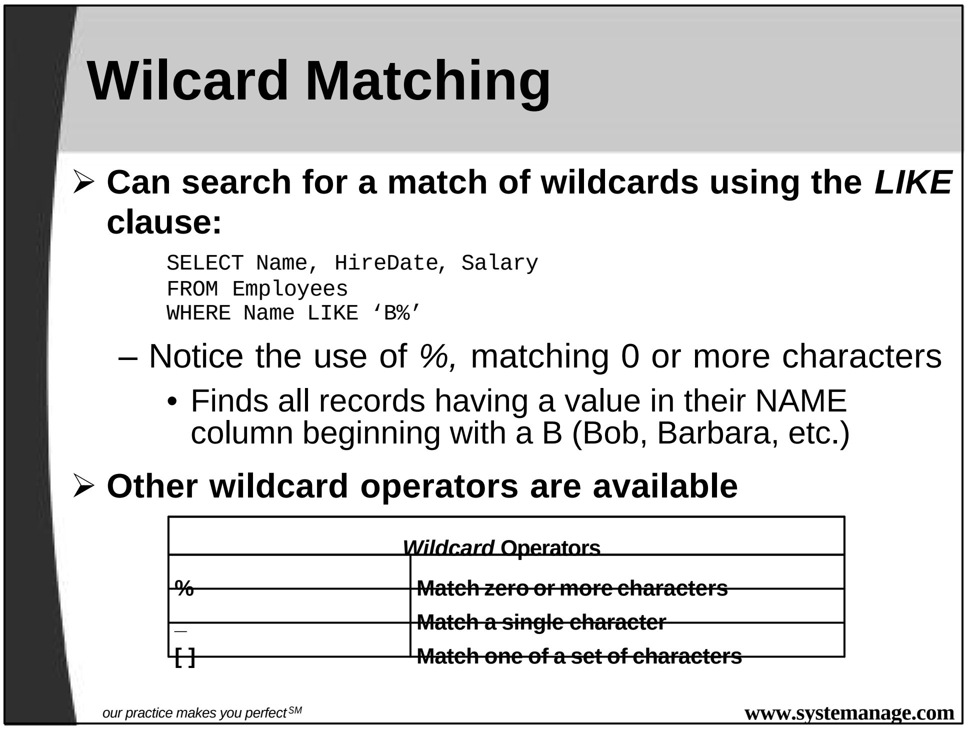
```
SELECT Name, HireDate, Salary
FROM Employees
WHERE EmpID IN (1,3,4)
```

  – Notice: values are separated with commas, enclosed within parentheses
  – If the column were string, would enclose each value in quotes
- This performs the equivalent of an “or” search
  – Finding records with EmpID 1 or 3 or 4



CF List Processing

- **Several ways to create/pass lists for the *IN* clause**
- **CF regards comma-separated values as a “list”**
  - Several list processing functions
  - Some variables may be available as lists, such as form variables for a checkbox form field
    - To put single-quotes around each value, use CF’s **ListQualify()** function
  - List of values of a given column in a previously executed query can be passed to an in clause, using the CF function **ValueList(*query.column*)**
    - See **QuotedValueList()** for columns of character datatype



Wilcard Matching

- **Can search for a match of wildcards using the *LIKE* clause:**

```
SELECT Name, HireDate, Salary
FROM Employees
WHERE Name LIKE ‘B%’
```

  - Notice the use of *%,* matching 0 or more characters
    - Finds all records having a value in their NAME column beginning with a B (Bob, Barbara, etc.)
- **Other wildcard operators are available**

Wildcard Operators	
%	Match zero or more characters
_	Match a single character
[ ]	Match one of a set of characters

More Wildcard Matching

- Wilcards can be used anywhere in string, not just at the beginning
  - To find records with name containing “ar”, like Charles, Arnold, Barbara, Karen, use:
    - `WHERE Name LIKE ‘%ar%’`
- Beware: wildcard matches are generally the slowest form of filtering
  - Use them with care
  - Particularly when pattern starts with wildcard
- Note, too, that the wildcard characters listed are ODBC wildcards, to be used when specifying SQL in CF
  - Curious: If % is used within Access query builder, will not match! It expects * instead. But if * is used within CF query passed to Access, it will not match!

Joining Multiple Filter Clauses

- **Can filter on multiple columns using AND and OR**
- **For instance, to find all Employees named Bob with a Salary above $20,000, use:**

```
SELECT Name, HireDate, Salary
FROM Employees
WHERE Name = ‘Bob’ AND Salary > 20000
```

- **To avoid ambiguity when using multiple filters, consider using parentheses to group criteria, as in:**

```
WHERE Name = ‘Bob’ AND (Salary > 20000 OR HighestGrade > 12)
```

Negating Filter Clauses

- **To negate a condition, use the *NOT* operator**
- **Examples:**

```
SELECT Name, HireDate, Salary
FROM Employees
WHERE NOT EmpID IN (3,5,7)
```

```
SELECT Name, HireDate, Salary
FROM Employees
WHERE TerminationDate IS NOT NULL
```

Sorting Data

- **To retrieve data in some particular sorted order, use the *ORDER BY* clause**

```
SELECT Name, HireDate, Salary
FROM Employees
ORDER BY Name
```

  - Creates resultset with records ordered by value of Name column
    - Of course, in this trivial example, would sort by first names. To sort by last names, would typically need an available LastName column
  - Can specify multiple, comma-separated columns
    - Data is sorted by the first column, then by the second if multiple rows have the same value for the first column
  - Data is sorted in ascending order by default
    - Can force descending order with DESC clause

Building Dynamic Queries

```
<CFQUERY DATASOURCE=“ProdPrsnl”
        NAME=“GetEmployees”>
   SELECT Name, HireDate, Salary
   FROM Employees
   <CFIF IsNumeric(Form.Salary)>
         WHERE Salary < #Form.Salary#
   </CFIF>
</CFQUERY>
```

- **Can build SQL dynamically at run time, using conditional statements and variables**
  - Powerful feature of CF, easier than other tools
- **ColdFusion processes the CF tags and variables before passing the resulting SQL to the database**

Performing Database Updates

- **SQL, despite its name suggesting it’s a “query language”, supports INSERT, UPDATE, DELETE**
- **ColdFusion also supports special CFINSERT and CFUPDATE tags (but no CFDELETE)**
  - Designed especially for causing all form data being passed to a template to be used for insert/update
  - While they are easier to use, they have several limitations and challenges
    - Can become cumbersome to use
    - Or may cause data loss or unexpected data transformation before insert/update
  - Many developers choose not to use the simpler tags and instead build the pure SQL clauses

INSERT Operations

```
INSERT INTO EMPLOYEES (Name, HireDate, Salary)
VALUES (‘Charles’,’09-05-2001’,20000)
```

- **The INSERT statement inserts one or more rows into a table, naming the table, columns & values**
  - Recall the importance of quoting strings used for columns with character datatypes
  - Must include all columns that do not permit nulls
  - Data can be inserted into (as well as updated in or deleted from) only one table at a time
- **There is an optional INSERT ... SELECT clause to insert multiple rows at once**
  - Inserts into the table the results of the SELECT clause

UPDATE Operations

```
UPDATE EMPLOYEES
SET TerminationDate = ‘09-05-2001’
WHERE EmpID = 1
```

- **The UPDATE statement updates data in one or more rows:**
  - naming the table to be updated, the rows to be affected, and the new values
  - Can update several columns, separating each column=value pair with a comma
- **Beware: if no WHERE clause is used, change is made to** ALL **rows in the table.**
  - Could be disastrous!
  - Could be intentional:

```
UPDATE PRODUCTS
SET PRICE = PRICE * 1.10
```

  - This would raise the price on all products by 10%

DELETE Operations

```
DELETE FROM EMPLOYEES
WHERE Terminationdate IS NOT NULL
```

- **The DELETE statement deletes one or more rows:**
  - naming the table to be processed and the rows to be affected
  - Notice that you do NOT name columns. Can only delete entire row.
- **Beware again: if no WHERE clause is used,** ALL **rows in the table are deleted!!**
  - Would be disastrous if unexpected!

Some Other Tidbits for You to Investigate

- **SELECT DISTINCT clause**
- **CFQUERY MAXROWS attribute**
  - Limits number of rows returned
- **CFOUTPUT's STARTROW and MAXROWS attributes**
  - Can specify starting point, max rows to process
- **CFLOOP also can loop over a query resultset**
- **Version 5's new CFQUERY CONNECTSTRING attribute**
- **Date processing in queries can be challenging**
  - Look into CF date functions, as well as DBMS-specific features for date handling

Where to Learn More

- **Version 5 CF manuals:**
  - Installing and Configuring ColdFusion Server
  - Developing ColdFusion Applications
  - CFML Reference
- **Books by Ben Forta:**
  - Teach Yourself SQL in 10 Minutes
  - Certified ColdFusion Developer Study Guide
  - ColdFusion Web Application Construction Kit
  - Advanced ColdFusion Development
- **Many other CF and SQL books available, including**
  - Practical SQL Handbook (new edition available)

*our practice makes you perfect*SM www.systemanage.com

Subjects of Next Two Seminars

- **Database 2: Slicing and Dicing Data in CF and SQL**
  - Working with Data in SQL Versus ColdFusion
  - Handling Distinct Column Values
  - Manipulating Data with SQL
  - Summarizing Data with SQL (Counts, Averages, etc.)
  - Grouping Data with SQL
  - Handling Nulls and Long Text
  - Cross-Referencing Tables (Joins)
- **Database 3: Improving Database Processing**
  - DB Performance & Scalability
    - Query Caching, BlockFactor, Indexes
  - DB Reliability
    - Constraints, Transactions, Bind Parameters, Triggers
  - DB Extensibility and Maintainability
    - Stored Procedures

*our practice makes you perfect*SM www.systemanage.com

Contact Information

Contact for follow-up issues

- **Email:** carehart@systemanage.com
- **Phone:** (301) 604-8399
- **Web:** www.systemanage.com

Also available for

- Training (custom or pre-written)
  - CF, DB, Jrun/J2EE, Javascript, wireless, and more
- Consulting (very short-term engagements)
  - best practices, architecture, setup, troubleshooting, etc.
- Developer Group Mentoring, and more

Q&A

?