SQL for Multiple DBMS

By Rob Kraft

Audience:	This document is for people developing applications that access data on different Database Management Systems (DBMS). When developing such applications, you will usually NOT want to take advantage of the enhanced features of a specific DBMS (such as stored procedures), because such features are difficult to maintain across multiple DBMS. Your goal is generally to minimize the customized work you will have to do for each DBMS. You want to minimize the customization of your database administration as well as minimize the customization of the SQL you write.
Background:	I created this document after learning first hand how much variation there is in SQL across DBMS while developing an application.
Your help:	I intend this document to never be complete. I hope that some readers will send me corrections where you see that they are needed. Also please send alternate solutions for the problematic areas. If you would like to include issues with additional DBMS, please send that information to me as well.
DBMS:	The Database Management Systems currently covered in this document include:
	Microsoft Access 97 (I believe all information applies to Access 2.0 and Access 95 as well).
	Microsoft SQL Server 6.5 (I believe all information applies to 4.2, 6.0, and 7.0 as well, except where noted).
	Oracle 8.0.4 (I believe all information applies to 7.3 versions as well).
	DB2 5.2
	Informix 7.2
About authors:	I, Rob Kraft, am the starting author of this document. My hope is to have many of you send contributions and become co-authors of this document. I work for Kraft Software Solutions, Inc. in Lee's Summit, MO. This document is on my company web site at: Kraft Software Solutions, Inc. My co-worker Jackie Rager contributed the information about DB2 and Informix. We hope that you benefit from our learning curve.
*4/29/99	New author - Dan Weinstein (dan.weinstein@bigfoot.com) has graciously submitted information on FoxPro databases, as well as clarifications on SQL Server information. Thanks to Dan from all of us!
Freeware:	We don't care what you do with this information as long as you give credit where credit is due, especially if you intend to make a profit from this information.
Date:	This version of the document is March 23, 1999.
Contacts:	You can contact the primary author of this document at robkraft@msn.com.

Table of Contents

* Denotes that the problem requires a different solution for different DBMSs AND requires writing different front-end SQL.
- Denotes that my recommendation deviates from ANSI standards.

String Delimiter

The string delimiter is used in SQL to let the DBMS know the beginning and end of a value that contains string, text, or character data. The name 'Kraft' is an example of such data.

Example:	SELECT * FROM CUSTOMERS WHERE LNAME = 'KRAFT'
Recommendation:	**Use SINGLE QUOTES as your string delimiter.**
ANSI SQL-92:	Single Quotes is the standard.

Access 97	Supports single quotes or double quotes.
MS SQL Server 6.5	Supports single quotes or double quotes (set driver option for double quotes).
Oracle 8.0.4	Supports single quotes only.
DB2 5.2	Supports single quotes only.
Informix 7.2	Supports single quotes or double quotes.
FoxPro	Supports single quotes or double quotes (also supports brackets [ ]).

According to ANSI standards, the double quotes are used to delineate column names. This allows column names with unusual characters, such as spaces, to be identified.

How to handle single quotes in your data

If you accept that a single quote is your string delimiter, you should determine how you will handle single quotes contained within your user data. The city name Lee's Summit is an example of such data.

Example:	SELECT * FROM CUSTOMERS WHERE CITY = 'LEE''S SUMMIT'
Recommendation:	**Use TWO SINGLE QUOTES to tell the DBMS to look for one single quote.**
ANSI SQL-92:	Using two single quotes is the standard.

This method of identifying single quotes in data is accepted by all of these DBMS.

Placing quotes around numeric data

Some DBMS allow you to place quotes around numeric data as well as string data. This is not the case for most DBMS.

Example:	SELECT * FROM CUSTOMERS WHERE ID = '15'
Recommendation:	**Do NOT put quotes around numeric.**
ANSI SQL-92:	Quotes are NOT allowed around numeric data.

Access 97	Quotes ARE allowed around numeric data.
MS SQL Server 6.5	Quotes are NOT allowed around numeric data.
Oracle 8.0.4	Quotes are NOT allowed around numeric data.
DB2 5.2	Quotes are NOT allowed around numeric data.
Informix 7.2	Quotes ARE allowed around numeric data.
FoxPro	Quotes are NOT allowed around numeric data.

Case on Column and Table Names

Are the table names and column names of the DBMS case sensitive? What should you do when writing SQL to access the tables and columns? It is obviously more work for programmers if they have to know the case of the table and column names when writing their SQL. Therefore I recommend that programmers always use upper case when writing SQL, and for case sensitive databases, the database administrator needs only to make sure upper case is used when creating all the tables.

Example:	CREATE TABLE CUSTOMERS (ID…)
Recommendation:	**Use UPPER CASE when writing SQL. Use UPPER CASE when creating tables on case-sensitive DBMS.**

ANSI SQL-92:	?
Access 97	Names are NOT case sensitive.
MS SQL Server 6.5	Names are NOT case sensitive.
Oracle 8.0.4	Names ARE case sensitive if the column names are enclosed in double quotes. Note that many ODBC driver place double quotes around column names by default. You may need to download the latest Oracle ODBC driver.
DB2 5.2	Names are NOT case sensitive.
Informix 7.2	Names are NOT case sensitive.
FoxPro	Names are NOT case sensitive.

Case in SQL keywords

What case should you use when typing the SQL keywords of SELECT, WHERE, FROM, etc? I am not aware of the case of the SQL statements being relevant, but I personally use all uppercase.

Example:	SELECT * FROM CUSTOMERS WHERE LNAME = 'KRAFT'
Recommendation:	**Use upper case for SQL keywords.**
ANSI SQL-92:	?

Case is irrelevant to all the DBMS covered here.

*Case sensitivity in data

Is the data stored and accessed in the DBMS case sensitive? Do the results of "Select * from customers where lname = 'Kraft'" return the same results as "Select * from customers where lname = 'KRAFT'"?

Example:	SELECT * FROM CUSTOMERS WHERE LNAME = 'Kraft'
	SELECT * FROM CUSTOMERS WHERE LNAME = 'KRAFT'

Recommendation:	**Be aware of the differences!!!**
ANSI SQL-92:	?

Access 97	The result sets are the same.
MS SQL Server 6.5	The result sets are the same (unless a case-sensitive sort order is chosen at installation of the DBMS).
Oracle 8.0.4	The result sets are NOT the same.
DB2 5.2	The result sets are NOT the same.
Informix 7.2	The result sets are NOT the same.
FoxPro	The result sets are NOT the same. (Has UPPER function similar to Oracle - see notes below).

Programmatically, the easiest solution is to store all data in upper case. Unfortunately, many users do not want all data to be displayed in upper case. This problem primarily affects where clauses. If you want to store mixed case data, here are some options for getting the case-INSENSITIVE results from your where clauses:

- Create an extra column in the table for each column that your user might search on. Store the uppercase version of the data in the extra column. This requires larger tables and also a different table structure for DBMS that do not support case-insensitivity.
- Use a function to convert the columns to upper case for comparison purposes. For example, in Oracle "Select lname, fname, state from customers where UPPER(lname) = 'KRAFT'". The drawbacks of this approach include: 1) your SQL is different for different DBMS, 2) you must figure out when to apply the UPPER function to your SQL, and 3) you lose performance and perhaps index utilization by converting the column value.

As I mentioned in the opening, if you can share a better solution, I would LOVE to hear about it.

*Wildcard characters for LIKE keyword

The LIKE keyword is used in conjunction with wildcard characters that may vary across DBMS. You may want to retrieve all customers with a last name that begins with a 'K', any value for the second character, has a third character of 'A', and any value for remaining characters.

Example:	SELECT * FROM CUSTOMERS WHERE LNAME LIKE = 'K_A%'
Recommendation:	**Use % for multiple characters, and _ for single characters.**
ANSI SQL-92:	% for multiple characters, and _ for single characters is the standard.
Access 97	* is used for multiple characters, and ? is used for a single character. However, if you connect to the database using MS OLE DB (or ADO), and you have applied MDAC 2.1 (released 2/99), it will used % and _ instead.
MS SQL Server 6.5	% for multiple characters, and _ for single characters.
Oracle 8.0.4	% for multiple characters, and _ for single characters.
DB2 5.2	% for multiple characters, and _ for single characters.
Informix 7.2	% for multiple characters, and _ for single characters. (The MATCHES clause used * and _).
FoxPro	% for multiple characters, and _ for single characters. (The setting of "SET ANSI ON | OFF" determines how comparisons are done if the lengths of the two strings are not equal - with ansi on they do not match, with ansi off, 'superman' is equal to 'super'.

-Syntax for Inner Join

An inner join is the most common method of joining two tables. An inner join between Customers and Orders would return a result set containing all the customers that had orders, but it would not include customers that had no orders or any orders that did not have a customer (see left join below for that).

Example:	SELECT * FROM CUSTOMERS, ORDERS WHERE CUSTOMERS.ID = ORDERS.CUSTID
Recommendation:	**Do NOT use ANSI standard join syntax unless you will only be using ANSI compliant DBMS.**
ANSI SQL-92:	Including the keyword INNER JOIN is the standard. SELECT * FROM CUSTOMERS INNER JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTID
Access 97	Supports INNER JOIN or join in WHERE clause.
MS SQL Server 6.5	Supports INNER JOIN or join in WHERE clause.
Oracle 8.0.4	Supports WHERE clause join only.
DB2 5.2	Supports INNER JOIN or join in WHERE clause.
Informix 7.2	Supports INNER JOIN or join in WHERE clause. (check on the syntax!)
FoxPro	Supports INNER JOIN or join in WHERE clause.

Any DBMS will support joining tables based on a condition in the where clause, but there are two drawbacks to doing this: 1) Spelling out the INNER JOIN syntax is more readable because you can easily distinguish the join clause from the parts of the where clause used to select which rows to return, and 2) Some DBMS provide incorrect results when joining in the where clause (particularly for queries involving null comparisons) because the DBMS does not know to perform the join criteria prior to the where criteria.

*Syntax for Left Join

A left join (or right join) retrieves ALL the rows from one table even if there is no match in the other table. A left join between Customers and Orders would return a result set containing ALL the customers, even those without orders (the order fields returned would be null), but it would not include any orders that did not have a customer.

Example:	SELECT * FROM CUSTOMERS LEFT JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTID
Recommendation:	**Never use a RIGHT JOIN - use a LEFT JOIN instead.** **Use ANSI LEFT JOIN syntax.**
ANSI SQL-92:	Including the keyword LEFT JOIN is the standard.
Access 97	Supports ONLY the ANSI notation for LEFT JOINs.
MS SQL Server 6.5	Supports the ANSI notation. Also has its own notation (Select * from Customers, Orders where customers.ID *= orders.custid). (Notice that *= means left join, =* means right join).
Oracle 8.0.4	Does NOT support ANSI LEFT JOIN. Has its own notation (Select * from Customers, Orders where customers.ID = orders.custid (+)). (Notice that the (+) follows the table name that will contain null data if there is no matches). *Chgd 4/22/98

DB2 5.2	Supports ONLY the ANSI notation for LEFT JOINs.
Informix 7.2	?
FoxPro	Supports ONLY the ANSI notation for LEFT JOINs.

A LEFT JOIN means that you will take ALL the rows from the first table listed (the one on the left) and only matching rows from the second table listed (the one on the right). The reason you don't need a RIGHT JOIN is that you can simple switch the order of the tables in your query to always make it a LEFT JOIN.
If you will only be using ODBC to access your DBMS, then you may be able to use ODBC extended SQL syntax. This applies particularly if you are going against just SQL Server and Oracle. The ODBC syntax allows you to write the SQL in a single format in your program. That format is:
SELECT * FROM {oj CUSTOMERS LEFT OUTER JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTID}
ODBC will then translate the left outer join into the correct format for the DBMS.

-AS clause

The AS clause is often used in two places, for substituting column names and for substituting table names. While the AS clause is supported by all DBMS here for column names, it is not for table names.

Example:	SELECT LNAME AS LASTNAME FROM CUSTOMERS C1 WHERE C1.ID = 51
Recommendation:	**Do NOT use the AS clause for table aliases, DO use the AS clause for column aliases**
ANSI SQL-92:	The standard is to use the AS clause for BOTH a column alias and a table alias.
Access 97	AS clause is accepted for both columns and tables, not required for tables.
MS SQL Server 6.5	AS clause is accepted for both columns and tables, not required for tables.
Oracle 8.0.4	AS clause is ONLY accepted for columns, not for table aliases.
DB2 5.2	AS clause is accepted for both columns and tables, not required for tables.
Informix 7.2	AS clause is ONLY accepted for columns, not for table aliases.
FoxPro	AS clause is accepted for both columns and tables, not required for tables.

*Concatenation operand

You may want to combine the data in multiple table columns to a single column in your resultset. This is done by concatenating the columns in the SQL. Unfortunately all DBMS may not support the same concatenation mechanism, therefore I recommend concatenating the columns in the calling program after receiving the resultset whenever possible.

Example:	SELECT LNAME || ', ' || FNAME FROM CUSTOMERS WHERE ID = 41
Recommendation:	**Avoid concatenation in SQL.**
ANSI SQL-92:	|| (two straight bars) is the ANSI standard concatenation character.
Access 97	Does NOT support the ANSI standard. Use & or + instead.
MS SQL Server 6.5	Does NOT support the ANSI standard. Use + instead.
Oracle 8.0.4	Supports the ANSI standard.
DB2 5.2	Supports the ANSI standard or the keyword CONCAT.
Informix 7.2	Supports the ANSI standard.
FoxPro	Does NOT support the ANSI standard. Use + instead.

Syntax of Delete keyword

Some DBMS allow a column list on a delete command, although it is irrelevant.

Example:	DELETE FROM CUSTOMERS WHERE LNAME = 'KRAFT'
Recommendation:	**Do not list * or columns in the delete command**
ANSI SQL-92:	DELETE FROM is the ANSI standard.
Access 97	Supports ANSI Standard. Also allows "DELETE * FROM …"

MS SQL Server 6.5	Supports ANSI Standard only.
Oracle 8.0.4	Supports ANSI Standard only.
DB2 5.2	?.
Informix 7.2	?.
FoxPro	Supports ANSI Standard only.

*Substring functions

Most DBMS offer a function to allow you to extract a range of characters from a string. Unfortunately the command for doing this varies across DBMS. All the functions here take 3 parameters: starting point, number of characters, column name.

Example:	SELECT substring(4,2,SSN) FROM CUSTOMERS WHERE ID = 41
Recommendation:	**Avoid obtaining substrings in SQL.**
ANSI SQL-92:	SUBSTRING.
Access 97	Use MID function.
MS SQL Server 6.5	Use SUBSTRING function.
Oracle 8.0.4	Use SUBSTR function.
DB2 5.2	Use SUBSTR function.
Informix 7.2	Use T1[n,m]?
FoxPro	Use SUBSTR(cExpression, nStartPosition [, nCharactersReturned])

*Trim functions

Most DBMS offer a function to allow you to trim spaces from a string. Unfortunately the command for doing this varies across DBMS.

Example:	SELECT trim(LNAME) FROM CUSTOMERS WHERE ID = 41
Recommendation:	**Avoid trimming strings in SQL.**
ANSI SQL-92:	TRIM.
Access 97	Use TRIM, LTRIM, or RTRIM.
MS SQL Server 6.5	Use LTRIM and RTRIM. In 6.5, empty strings are returned as a single space. In 7.0, empty strings will be returned as empty strings (no single space).
Oracle 8.0.4	Use LTRIM and RTRIM.
DB2 5.2	Use LTRIM and RTRIM.
Informix 7.2	Use TRIM (lead, trail, both).
FoxPro	Use TRIM, LTRIM, or RTRIM, or Alltrim(). Alltrim = ltrim(rtrim()). Trim() = rtrim().

Distinct clause - eliminate duplicate rows

Most DBMS offer a function to allow you to eliminate duplicate rows from the resultset. This may cause the data to be sorted on some DBMS.

Example:	SELECT DISTINCT LNAME FROM CUSTOMERS
Recommendation:	**Use DISTINCT to eliminate duplicate rows.**
ANSI SQL-92:	DISTINCT is the ANSI standard.
Access 97	DISTINCT is supported. An alias is DISTINCTROW.
MS SQL Server 6.5	DISTINCT is supported.
Oracle 8.0.4	DISTINCT is supported.
DB2 5.2	DISTINCT is supported.
Informix 7.2	DISTINCT is supported. An alias is UNIQUE.
FoxPro	DISTINCT is supported.

Support for automatic counters, identity columns, autonumbers

An automatic counter is an ID generated by the DBMS when a row is inserted. The column name of the counter is generally not allowed on the insert, and the distinct value placed in the counter column is maintained by the DBMS. Not all DBMS support a counter that is as easy to use as that provided with Microsoft Access, but the concept can be simulated.

Example:	INSERT INTO CUSTOMERS VALUES()
Recommendation:	**Use counters but know how to simulate them in each DBMS so that no changes are required to your programs that make calls to the DBMS.**
ANSI SQL-92:	COUNTER().
Access 97	Supported - Autonumber. Begins at 1 and increments by 1. Access 2000 will support the option to begin at any integral value and increment or decrement by any integral value.
MS SQL Server 6.5	Supported - Identity column. Start at any integral value and increment or decrement by any integral value.
Oracle 8.0.4	Not supported. Oracle provides an independent counter object called a SEQUENCE. You must create a SEQUENCE for each counter, then create an insert trigger for each table to call the SEQUENCE and apply the next value to the column in the table.
DB2 5.2	Not supported. You will need to create your own table to track the next number to be assigned. Then create an insert trigger for each table to get the next value from your counter table, and update the counter table.
Informix 7.2	Supported - Identity column. Start at any integral value and increment or decrement by any integral value.
FoxPro	Not supported. You will need to create a stored proc to emulate this functionality.

Special character support in column names

Each DBMS supports non-alphanumeric characters in the names of tables and columns. You will want to select names that are supported across all DBMSs, and this limits your special character options.

Example:	CREATE TABLE CUSTOMERS (F_NAME VARCHAR2(20), PHONE# CHAR(15))	
Recommendation:	**Only use the underscore and alphanumerics in table and column names. Start all table and column names with an alphabetic letter.**	
ANSI SQL-92:	?	
Access 97	Any character except:	.!`[]
MS SQL Server 6.5	Only allows:	_ $ #
Oracle 8.0.4		
DB2 5.2		
Informix 7.2		
FoxPro	Only allows:	_

*Convert to string function

Most DBMS offer a function to allow you to convert a number to a string. Unfortunately the command for doing this varies across DBMS.

Example:	SELECT STR(BLDG_NO) & ADDRESS AS FULLADDRESS FROM CUSTOMERS
Recommendation:	**Avoid convert to string functions in SQL.**
ANSI SQL-92:	CAST.
Access 97	Use STR().
MS SQL Server 6.5	Use STR() or CONVERT().
Oracle 8.0.4	Use TO_CHAR()
DB2 5.2	
Informix 7.2	
FoxPro	Use STR()

*Dealing with Dates and Times

This is one of the most complex areas to manage. I suggest you resolve how this will be handled first in your development. The question is, how do I let the DBMS know I am providing a date value, and how to I retrieve rows matching a particular date value. Suppose you want to retrieve all rows modified on 2/27/1999 before 8:00. While the query does not sound unreasonable, you may find it very challenging on different servers. For this first case, let's assume the dates and times are stored in different columns.

Example:	SELECT * FROM CUSTOMERS WHERE MOD_DT = '2/27/1999' AND MOD_TM < '8:00 AM'
Recommendation:	**Write a function in your calling program to format the date fields.** **Example: strSQL = "SELECT * FROM CUSTOMERS WHERE MOD_DT = " & fnFormatDate (dtBegin, "Oracle")**
ANSI SQL-92:	?
Access 97	Use pound signs (#) to denote date and time data. WHERE MOD_DT =#'2/27/1999# AND MOD_TM < #8:00 AM# Access understands many date formats: mm/dd/yy, mm-dd-yy, yy-dd-mm, mm.dd.yy, etc.
MS SQL Server 6.5	Use single quotes (') to denote date and time data. WHERE MOD_DT = '2/27/1999' AND MOD_TM < '8:00 AM' SQL Server understands many date formats: mm/dd/yy, mm-dd-yy, yy-dd-mm, mm.dd.yy, etc.
Oracle 8.0.4	By default, Oracle only recognizes dd-mmm-yy. You can change the default format to something else, such as mm/dd/yyyy by issuing "ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYY';" However, times stored in date fields expect the same format and therefore must by converted by the SQL. Use single quotes (') to denote date and time data. WHERE MOD_DT = '2/27/1999' AND TO_DATE(TO_CHAR(MOD_TM, ‘HH:MI:SS PM’), 'HH:MI:SS PM') < '8:00 AM'
DB2 5.2	Use single quotes (') to denote date and time data. WHERE MOD_DT = '2/27/1999' AND MOD_TM < '8:00 AM' DB2 understands many date formats: mm/dd/yy, mm-dd-yy, yy-dd-mm, mm.dd.yy, etc.
Informix 7.2	Use single quotes (') to denote date and time data. WHERE MOD_DT = '2/27/1999' AND MOD_TM < '8:00 AM' Informix understands many date formats: mm/dd/yy, mm-dd-yy, yy-dd-mm, mm.dd.yy, etc.
FoxPro	Use {} to denote dates. WHERE MOD_DT = {2/27/1999} Also you can use the convert to date function: WHERE MOD_DT = ctod('2/27/1999')

This is the area I had the most difficulty with for Oracle, I would greatly love to hear about some better approaches.
For any of these servers, if date and times are stored within the same datetime column, another complexity usually arises. In trying to obtain all rows modified on 2/27/1999, you may be tempted to code WHERE MOD_DTTM = '2/27/1999'. However, since no time was specified, the DBMS assumes you also mean where time is equal to 12:00am and your result set is empty. For this scenario, I recommend this: WHERE MOD_DTTM >= '2/27/1999' AND MOD_DTTM < '2/28/1999'. Notice that in dealing with dates and times, it is almost a certainty that you will need to code your front end (but hopefully your middle tier) differently for each DBMS.
If you will only be using ODBC to access your DBMS, then you may be able to use ODBC extended SQL syntax. This applies particularly if you are going against just SQL Server and Oracle. The ODBC syntax allows you to write the SQL in a single format in your program. That format is:
SELECT * FROM CUSTOMERS WHERE MOD_DT < {D '1999-02-27'}
ODBC will then translate the date into the correct format for the DBMS.

*Retrieving the current date

Most DBMS offer a function to retrieve the current date. Unfortunately the command for doing this varies across DBMS.

Example:	SELECT CURRENT_DATE()
Recommendation:	**Retrieve the current date from the network OS instead of the DBMS.**
ANSI SQL-92:	?
Access 97	SELECT DATE() FROM TABLE WHERE 1 > 0. You must provide the name of any existing table.
MS SQL Server 6.5	SELECT GETDATE()

Oracle 8.0.4	SELECT SYSDATE FROM DUAL
DB2 5.2	?
Informix 7.2	SELECT DATE(CURRENT)
FoxPro	SELECT DATE() FROM anytablename. Also you can just 'return date()'.

Constraints can be used to enforce referential integrity

Most DBMS support ANSI standard constraints to manage Primary Key/Foreign Key relationships between tables (referential integrity). This form of integrity is usually more robust and less overhead than any others.

Example:	ALTER TABLE ORDERS ADD CONSTRAINT FK1 FOREIGN KEY (CUSTID) REFERENCES CUSTOMERS (ID)
Recommendation:	**Use PK and FK to enforce referential integrity.**
ANSI SQL-92:	Supported - syntax as shown above is very similar across DBMS.

Access 97	Supported
MS SQL Server 6.5	Supported
Oracle 8.0.4	Supported
DB2 5.2	Supported
Informix 7.2	Supported
FoxPro	Supported

Cascade Deletes

Some DBMS allow your database relationships to cascade deletes that occur. For example, if a customer is deleted, and cascade delete is enable for the customer to orders relationship, then the orders for that customer will also be automatically deleted. If cascade delete is not enabled, the delete for the customer will fail if orders for the customer exist. Cascading of a delete is a business rule and should probably be handled in the business objects, however DBMS do such a good job of it cascading deletes that it makes good sense to use the DBMS to perform the action. The drawback is that different approaches are required on each DBMS to create a cascade delete, but fortunately it does not affect the SQL in your calling programs. Many DBMS allow you to create a CASCADE DELETE via a CONSTRAINT, or a TRIGGER. In every case I would recommend a CONSTRAINT over a TRIGGER. They are easier to maintain and are more efficient.

Example:	DELETE FROM ORDERS WHERE CUSTID = 41
	DELETE FROM CUSTOMERS WHERE ID = 41

Recommendation:	**Use business objects to enforce cascade deletes.**
ANSI SQL-92:	The CASCADE DELETE constraint is defined by ANSI.

Access 97	Cascade deleted can be implemented by constraints - a property of the relationship definition.
MS SQL Server 6.5	Cascade delete can only be implemented by triggers. This is true in SQL Server 7.0 as well. When applying a cascade delete trigger you will have to remove the PK/FK constraint or else the PK/FK constraint will prevent the delete trigger first. If you remove the PK/FK constraints, you will need to add INSERT and UPDATE triggers to enforce referential integrity.
Oracle 8.0.4	Cascade deleted can be implemented by constraints - a property of the relationship definition. They can also be implemented by triggers.
DB2 5.2	?
Informix 7.2	?
FoxPro	Cascade can only be achieved with triggers. It is NOT recommended that you rely on the relational integrity wizard to write these for you.

Referencing table names that have owners

On many DBMS the same table name can be created by multiple people because the DBMS distinguishes each table by the table owner name. This can complicate the writing of SQL if you have to account for the owner of the table name. You definitely want to have all your tables owned by the same account to make it unnecessary to adjust your SQL.

Example:	SELECT * FROM ROB.CUSTOMERS WHERE LNAME = 'KRAFT'
Recommendation:	**Do not specify the table OWNER name for every column/table listed in the SQL.**
ANSI SQL-92:	not applicable
Access 97	Table owner names are not applicable.
MS SQL Server 6.5	Make dbo the owner of all tables.
Oracle 8.0.4	Have a single account create all the tables. Create a SYNONYM for accessing the tables without specifying the owner account name.
DB2 5.2	?
Informix 7.2	?
FoxPro	Table owner names are not applicable.

Symbol for Not Greater Than and Not Equal To

Some DBMS support multiple ways of specifying "Not Greater Than", or "Not Equal To". Fortunately the ANSI standard works on all the DBMS mentioned here.

Example:	SELECT * FROM CUSTOMERS WHERE ID <= 41 AND ID <> 15
Recommendation:	**Use <= for "Not Greater Than" and <> for "Not Equal To".**
ANSI SQL-92:	Use <= for "Not Greater Than" and <> for "Not Equal To".
Access 97	<= and <>
MS SQL Server 6.5	<= and <> Also supports !> and !=
Oracle 8.0.4	<= and <> Also supports !=
DB2 5.2	<= and <>
Informix 7.2	<= and <> Also supports !=
FoxPro	<= and <> Also supports != and # (for not equal), and !>

Symbols for NULL Testing

Some DBMS support multiple ways of specifying NULL or NOT NULL for comparisons. Fortunately the ANSI standard works on all the DBMS mentioned here.

Example:	SELECT * FROM CUSTOMERS WHERE FNAME IS NULL
Recommendation:	**Use IS NULL and IS NOT NULL**
ANSI SQL-92:	IS NULL and IS NOT NULL
Access 97	IS NULL and IS NOT NULL
MS SQL Server 6.5	IS NULL and IS NOT NULL. Also supports = NULL and <> NULL.
Oracle 8.0.4	IS NULL and IS NOT NULL
DB2 5.2	
Informix 7.2	IS NULL and IS NOT NULL
FoxPro	IS NULL and IS NOT NULL. Also supports = NULL and <> NULL.

Datatype comparisons

You also need to be cautious in choosing column datatypes. When writing applications that will run on multiple DBMS you need to select datatypes that are supported across all the DBMS. Here are some of the datatypes I would recommend.

Datatypes	**Access 97**	**Microsoft SQL Server 6.5**	**Oracle 8.04**	**DB2 7.2**	**Informix 7.2**	**FoxPro**
Integer (-2 billion to +2 billion)	Long Integer	int	Number(10,0)	Integer	Integer	Integer

SmallInt (-32000 to +32000)	Integer	smallint	Number(5,0)	Smallint	Smallint	Number(
VarChar (Text data like names)	Text	varchar	varchar2	Varchar	Varchar	Memo
Bit (1 or 0)	Yes/No	bit	Number(1,0)	?	?	Logical
Dates and Times	Date/Time	datetime	Date	Timestamp	Datetime	Date and
Floating Point Numbers	Double	float or decimal	Double Precision	Double	Float or decimal(p)	Float
Currency	Currency	money	Number(9,2)	Decimal(n,2)	money	Currency
Default values for bit?	Default of Yes or No (but use 1 or 0)	Default of 1 or 0	Default of 1 or 0	Default of 1 or 0	Default of 1 or 0	Default c false
BLOBs/Memos/ Large Datatypes	Memo	text	Long	Blob/clob/dbclob	Blob/Text	Memo
	Many per table	Many per table	1 per table (at end of table???)	?	?	Many pe

Notes:
1. For SQL Server, and perhaps other DBMS, bit datatypes cannot be NULL (though they can be NULL in SQL 7.0). I recommend making bit datatypes a required field in all DBMS, and applying a default value to the field in the DBMS.
2. Oracle only allows one LONG field per table (MEMO, TEXT, BLOB). For this reason, and for performance reasons, I recommend creating a separate two-column table to store in one column the BLOB, and in the other column the ID used to retrieve it.

DBMS Limitations

You also need to be aware of the limitations of each DBMS. Take particular note of the maximum length of column and table names, because if you use a 34 character table name on one DBMS, the name may be too long for the other DBMS. Along with that, try very hard to avoid any table or column names that could be a keyword on any DBMS.

Limitations	**Recommendation**	**Access 97**	**Microsoft SQL Server 6.5**	**Oracle 8.04**	**DB2 7.2**	**Informix 7.2**
Column name lengths	Maximum of 14	64	30	?	18	18
Number of fields in a table		255	250 (1024 in 7.0)	1000	500	
Number of characters in a record (excluding Memo and OLE Object fields)		2000	1962 (8060 in SQL 7.0)		4005	
Table size		1 gigabyte	1 terabyte (1,000,000 TB 7)		64 * 999 gigabytes	
Number of characters in a Text field		255	255 (8000 in SQL 7.0)	char = 2k, varchar = 4k	char=254 varchar=4k longvarchar=32k	255
Number of characters in a Memo field		65,535 or 1 gigabyte	2 gigabytes	4 gigabytes	2 gigabytes	2 gigabytes
Number of indexes in a table		32	250		32767	
Number of fields in an index		10	16		16	
Longest SQL Stmt	2k - use insert followed by	64k	128k	64k	32767	

	update for more					
Most columns in select list		255	4096	250?	500	

Miscellaneous, but IMPORTANT notes

1. The result of a concatenation with a NULL value may vary across DBMS. In some the result will return a NULL for the entire value, in others it will return the non-null concatenated values. This behavior changed for SQL Server between versions 6.5 and 7.0.
2. The driver you use to access the DBMS may alter the SQL before it gets there. This is particularly true if you pass the SQL through ODBC.
3. The SQL Syntax for Crystal Reports and other reporting engines may be drastically different than what you send directly from your program to the DBMS. You should evaluate other DBMS interfaces like reporting packages that you will use when designing your application.
4. Microsoft's new data access approach OLE/DB (ADO) may help or hurt your efforts. You will have to see what form of SQL the OLE/DB interface expects for each DBMS. An important note for Microsoft Access developers is that the newest version of Microsoft's data access approach (MDAC 2.1) changes the LIKE wildcards from * and ? to the ANSI standard % and _.
5. Using Linked tables in Access is one mechanism for avoiding having to write different SQL for different DBMS. However, contrary to what you may read, this is not always a fast approach. For some types of queries it is faster than native connections to the back-end SQL Server or Oracle server, but for others, particularly joins, it is intolerably long. To test this, create links to two tables on your back-end server that are related and have a lot of rows. Write an SQL that joins the tables and returns all rows. Run the query and issue a MoveLast command. Go to lunch. Come back and check the results.
6. *Added 4/22/98. Oracle does not allow a subselect of an INSERT command to contain LONG datatypes (memo datatypes in Access).