R&R Report Writer *Technical Bulletin*

Checking Database Integrity

Product:	ARPEGGIO™ R&R ReportWriter®	**NIC:**	N.A.
Version:	All	**Interface:**	N.A.
Host:	N.A.	**Oper Sys:**	Microsoft® Windows® Microsoft Windows 95 Microsoft Windows NT®

Summary

Here is a tip every database user will find handy for checking the integrity of a database. Bad data seems to creep into databases despite attempts to validate data during data entry. We'll discuss the sources of bad data and suggest a reporting technique you can use to expose bad data. The resulting reports can help you find and correct database integrity problems.

Make a List of Fields

This technique requires that you first make a written or mental list of fields you want to check and their associated integrity rules. Use your experience with an application to come up with a reasonable battery of tests. For example, you may want to ensure that a shipment date never precedes the order date, or that an unpaid invoice is never older than 120 days.

Types of Bad Data

Bad data comes in three flavors: missing, invalid, and inconsistent. Using the fields CITY and STATE as an example, a blank state would be an example of missing data, the state XY would be an example of invalid data, and the state NY together with the city *Boston* would be an example of inconsistent data.

There are four sources of bad data:

1. The correct data is not known at the time of data entry.
2. The application designer doesn't or can't provide data entry validation.
3. The user is able to circumvent the application.
4. There may be unknown bugs in the software.

© 1999 Liveware Publishing Inc.

In this situation we'll use a mailing list to illustrate the sources of bad data, the tests you can perform to identify problems, and the way to design a report so that you can make use of the results. We've purposely kept this example simple; however, the same technique applies even if you have complex database integrity rules.

Database Integrity Checks

Figure 1 shows a sample database integrity check report. The record number appears in the first column and serves as a key for making corrections to the database. The three integrity checks illustrate three of the four different types of bad data we mentioned above. (We'll talk about the fourth type, software bugs, later.) Note that only records containing bad data are listed, which is why there are gaps between record numbers.

Database Integrity Check

Record Number	Missing Zip Code	Invalid State	Inconsistent Data
12			
143	X		
210	X	X	
340			X
419	X		
539			
593		X	X
653			

Figure 1. Database Integrity Check Report

The report layout is very simple. It consists of a page header with the report name and column headings, plus one record line that contains the fields that identify the record number and the integrity checks that failed. We'll explain these fields below.

The record number is produced using a calculated field. Let's assume the mailing list database is named MAILLIST. Use the Calculations/Calculated Field command to define the record number as follows.

```
RecordNo = RECNO(maillist)
```

Checking for Missing Data

The first integrity check flags records that have missing ZIP codes. Perhaps the ZIP codes were not known at the time of data entry, but before doing a mailing you want to enter them.

The following calculated field is used to place an X in the column labeled *Missing ZIP Code* when the ZIP code (a character field) is blank.

```
BadZIP = IIF(.NOT.ZIP,"X","")
```

The BadZIP expression says "if there's nothing in the ZIP field, return an X, otherwise return a blank."

Checking for Invalid Data

The second integrity check flags records that have invalid state codes. Perhaps you forgot to, or weren't able to, provide a data entry validation for the two letter state abbreviations, but you'd like to correct them before doing your mailing.

The following calculated field is used to place an X in the column labeled *Invalid State* when the STATE field contains an invalid state abbreviation.

```
BadSTATE = IIF(.NOT.(STATE$
    "AL,AK,AZ,AR,CA,CO,CT,DE,DC,FL,GA,HI,ID,IL,IN,IA,KS,KY,
    LA,ME,MD,MA,MI,MN,MS,MO,MT,NE,NV,NH,NJ,NM,NY,NC,ND,
    OH,OK,OR,PA,RI,SC,SD,TN,TX,UT,VT,VA,WA,WV,WI,WY"),
    "X","")
```

The expression is broken into several lines for readability, although you'd enter it in R&R all on one line. The expression says "if the state code is not one of the 5O valid abbreviations, return an X, otherwise return a blank."

Each record in our mailing list contains two dates: the date the record was added to the file, CreateDATE, and the date the record was last updated, ModifyDATE. The third integrity check flags records where the creation date does not precede the date of last update. This situation could arise if a user circumvents the normal data entry procedure and changes one or both of these dates.

Checking for Inconsistent Data

The following calculated field is used to place an X in the column labeled *Inconsistent Dates* when the creation date is more recent than the date last modified.

```
BadDATES = IIF(CreateDATE>ModifyDATE,"X","")
```

This expressions says "if the creation date comes after the modification date, return an X, otherwise return a blank."

Querying Records that Fail Integrity Checks

The final stage in creating a database integrity check report is to create a query that selects only records that fail one or more integrity checks; there is no need to print all records in the

database. Using the Database/Filter command, specify a selection of all records where any of the integrity check fields contains an X. Connect the individual filter clauses with the OR connector as follows.

BadZip is equal to "X" or BadSTATE is equal to "X" or BadDATES is equal to "X"

In more sophisticated relational applications, you may need to perform cross-file integrity checks. For example, we have a field in our mailing list that stores the total number of orders placed by each person. This number is incremented by our data entry program each time an order is added to the order database.

Periodically, we verify that the number stored in the mailing list corresponds to the actual number of orders in the order file to insure there aren't any bugs in the data entry program. Since there could be many orders per customer, we'd need to use a scan relation and move the record line to the group footer area.

The example integrity checks presented in this example, while simplistic, illustrate the basic types of integrity check you can apply in your own applications. It is a good idea to run your database integrity check reports on a regular basis as a preventative measure.

All trademarks are the property of their respective owners. The information contained in this technical bulletin is subject to change without notice. Liveware Publishing Inc. provides this information “as is” without warranty of any kind, either expressed or implied, but not limited to the implied warranty of merchantability and fitness for a particular purpose. Liveware Publishing may improve or change the product at any time without further notice; this document does not represent a commitment on the part of Liveware Publishing. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the licensing agreement.