

# The Decimal datatype White Paper

Published: August 1998 (with revisions)

---

## The Decimal datatype - It's not just for COBOL anymore!

---

### Contents

Preface	1
Overview	1
Example	2
Summary	4
About the Author	5
Let Us Help You Succeed!	5

---

### Preface

*“The want of MONEY is the root of all evil”.* The saying commonly credited to Mark Twain, was actually coined by the late English author, Samuel Butler. Do you want to know about the lurking evil in the MONEY datatype in Ingres? And what to do about it – Read on!

### Overview

The DECIMAL datatype was introduced in OpenIngres 1.x several years ago. This was a welcome introduction to COBOL developers since they now had a native datatype. Besides being an easy to use datatype for COBOL there are other benefits to using the decimal datatype over other datatypes, specifically the MONEY datatype.

The MONEY datatype is very useful but does have a few drawbacks. It can be somewhat limited in its range (-999,999,999.99 to 999,999,999.99), depending on your application (and currency). It also has issues relative to rounding, which will be discussed further below. NOTE: For more information on the MONEY or DECIMAL datatypes, please refer to the OpenIngres SQL Reference Guide.

Using the DECIMAL datatype with the default range of the MONEY datatype will take the same amount of space (internally) per column (i.e., 8 bytes when not nullable), when declared as "decimal(14,2)". In this example the *precision* (total number of digits) is 14 and the *scale* (number of digits to the right of the decimal point) is 2. The DECIMAL datatype can be created to have a maximum precision of 31, therefore allowing the storage of much larger currency values. Using the "money()" function it is still possible to display the currency symbol (provided the amount does not exceed the valid range for the money datatype), so there really is no loss in functionality by making this type of change.

## Example

Below is an example that shows the true benefit of the DECIMAL datatype over the MONEY datatype. As you will see a table is created that contains two columns; one using the money datatype (mny\_col) and the other using the decimal datatype (dec\_col). Nine rows of data, each with identical values for each column, are added to this table. Selection of that data shows identical values. Next there is a query to calculate tax using two different methods (one simulates the computation by "line item", while the other simulates the computation on the entire order). Both are valid computations, but as you can see the calculations on the MONEY datatype yield two different results (where one was incorrect due to rounding). You will also see that the calculations for the DECIMAL datatype yielded the same results.

The final query is an example of casting the DECIMAL column (written the "wrong way" if we were using the money datatype) to a MONEY datatype for display purposes. **This is a good point to keep in mind as you review your queries and / or schema in the future!**

```
INGRES TERMINAL MONITOR Copyright (c) 1981, 1997 Computer
Associates Intl, Inc.
OpenIngres SPARC SOLARIS Version OI 2.0/9712
(su4.us5/00) login
Continue
* * * * /* SQL Startup File */
create table roundingtest (
mny_col money not null not default,
dec_col decimal (14,2) not null not default);
Executing . . .
Continue
* * * *
insert into roundingtest (mny_col, dec_col)
values (10, 10);
Executing . . .
(1 row)
continue
* * * *
insert into roundingtest (mny_col, dec_col)
values (1, 1);
Executing . . .
(1 row)
continue
```

```
* * * *
insert into roundingtest (mny_col, dec_col)
values (1, 1);
Executing . . .
(1 row)
continue
* * * *
insert into roundingtest (mny_col, dec_col)
values (1, 1);
Executing . . .
(1 row)
continue
* * * *
insert into roundingtest (mny_col, dec_col)
values (1, 1);
Executing . . .
(1 row)
continue
* * * *
insert into roundingtest (mny_col, dec_col)
values (1, 1);
Executing . . .
(1 row)
continue
* * * *
insert into roundingtest (mny_col, dec_col)
values (.01, .01);
Executing . . .
(1 row)
continue
* * * *
insert into roundingtest (mny_col, dec_col)
values (.001, .001);
Executing . . .
(1 row)
continue
* * *
select * from roundingtest;
Executing . . .
```

```
+-----+-----+
| mny_col | dec_col |
+-----+-----+
| $10.00 | 10.00  |
| $1.00  | 1.00   |
| $1.00  | 1.00   |
```

```

| $1.00 | 1.00 |
| $1.00 | 1.00 |
| $1.00 | 1.00 |
| $1.00 | 1.00 |
| $0.01 | 0.01 |
| $0.00 | 0.00 |
+-----+-----+
(9 rows)
continue
* * * * *
select sum(mny_col) * .065 as mny_tax_meth1,
sum(mny_col * .065) as mny_tax_meth2,
sum(dec_col) * .065 as dec_tax_meth1,
sum(dec_col * .065) as dec_tax_meth2
from roundingtest;

```

Executing . . .

```

+-----+-----+-----+-----+
|mny_tax_meth1 |mny_tax_meth2 |dec_tax_meth1|dec_tax_meth2 |
+-----+-----+-----+-----+
|          $1.04 |          $1.07 |    1.04065  |    1.04065  |
+-----+-----+-----+-----+

```

(1 row)

continue

\* \* \* \*

```

select money(sum(dec_col) * .065) as mny_tax
from roundingtest;

```

Executing . . .

```

+-----+
| mny_tax |
+-----+
|   $1.04 |
+-----+

```

(1 row)

## Summary

Since it is not possible to control the way every query is written it is possible to introduce data integrity or other business issues if a query is written "the wrong way" using the MONEY datatype. Fortunately there is no wrong way to write the query using the DECIMAL datatype. The business importance of this simply cannot be overstated!

This article was originally published in the Fall, 1998 North American Ingres Users Association (NAIUA) newsletter.

## About the Author

Chip Nickolett is the President of Comprehensive Solutions. He has been using Ingres since 1986. He was a Senior Consultant with Ingres, was a top-ranked Ingres consultant at Computer Associates, and is a past-President of the North American Ingres Users Association (NAIUA). He has been a very active participant in the Ingres community most of his career.

## Let Us Help You Succeed!

Call today to discuss ways that Comprehensive Solutions can help your organization save money and achieve better results with your IT projects. We provide the *confidence* that you want and deliver the *results* that you need.

[View our "Ingres Support" Brochure](#)

[Back to White Papers](#)

[Back to Services](#)

Comprehensive Solutions  
4040 N. Calhoun Road  
Suite 105  
Brookfield, WI 53005  
U.S.A.

Phone: (262) 544-9954

Fax: (262) 544-1236

Copyright © 1998-2008 Comprehensive Consulting Solutions, Inc.

All Rights Reserved

No part of this document may be copied without the express written permission of Comprehensive Consulting Solutions, Inc., 4040 N. Calhoun Rd., Suite 105, Brookfield, WI 53005.

This document is provided for informational purposes only, and the information herein is subject to change without notice. Please report any errors herein to Comprehensive Consulting Solutions. Comprehensive Consulting Solutions, Inc. does not provide any warranties covering and specifically disclaims any liability in connection with this document.

All product names referenced herein are trademarks of their respective companies. Use of these names should not be regarded as affecting the validity of any trademark or service mark.