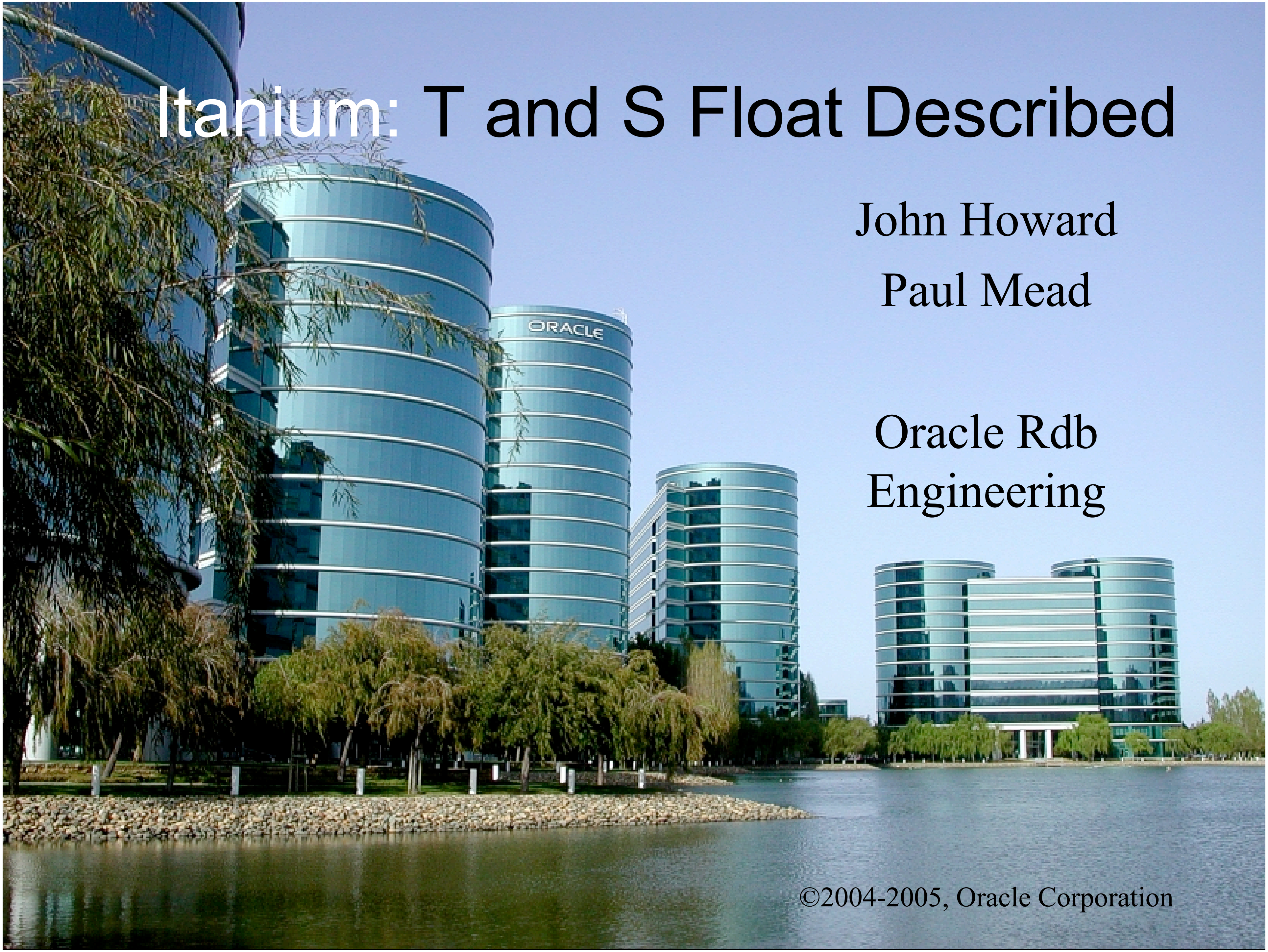


Itanium: T and S Float Described

John Howard
Paul Mead

Oracle Rdb
Engineering

Agenda

- Floating point explained
- VAX/Alpha vs. IEEE floating point types
- Rdb floating point support
- IEEE floating point features
- Porting your application
  - Precompiled SQL
  - SQL Module Language
  - Dynamic SQL

What is Floating Point?

- Approximate vales in scientific notation:
  - For example: 12340000 = 1.234×10^7 with these parts:
    - ☒Significand (a.k.a. fraction) - 1.234×10^7
    - ☒Base - 1.234×10^7
    - ☒Exponent (a.k.a. mantissa) - 1.234×10^7
  - Both the significand and the exponent have a sign
    - ☒Significand sign is the sign of the number
    - ☒Exponent sign scales the number (e.g. $10^{-2} = 1/100$)
- Useful for representing real numbers
  - Standard for scientific applications and math libraries
  - Rarely used in most commercial applications
  - SQL datatypes: REAL, FLOAT, DOUBLE PRECISION

Floating Point in Computers

- Floating point datatypes have a certain layout with a set of bits devoted to the significand, exponent, and their signs.
  - Finite bits mean datatypes have finite ranges
  - Base is assumed and is usually two
- Floating point arithmetic can be supported in software but is usually done in hardware.
  - VAX, Alpha and Itanium all have different floating point types and hardware.
  - The IEEE 754 standard allows machines to have compatible formats and consistent results across machines.

Floating Point is Approximate!

- Arithmetic is subject to several error sources:
  - Range, for example, with 32 bit floating point: $1.0*10^8 + 1.0 = 1.0*10^8$, as if nothing happened!
  - Rounding
  - Representation, for example: 0.1 decimal is an irrational number expressed as a base two floating point number: $1.1001100\ldots*2^{-4}$
- Applications must be designed to minimize errors:
  - Adding 1.0 to $1.0*10^8$ for 10^8 repetitions yields $1.0*10^8$
  - But $1.0*10^8 + (1.0*10^8 \text{ x } 1.0)$ yields $2.0*10^8$

VAX/Alpha and IEEE FP Types

Type	Bits (exponent, significand)	Hardware Support	Decimal Range	Precision (digits in significand)
F Float	32 (8, 23)	VAX, Alpha	$.29*10^{-38}$ to $1.7*10^{38}$	6 decimal digits
S Float	32 (8, 24)	Alpha, Itanium	$1.18*10^{-38}$ to $3.40*10^{38}$	6 decimal digits
D Float	64 (8, 55*)	VAX, Alpha*	$.29*10^{-38}$ to $1.7*10^{38}$	16 decimal digits*
G Float	64 (11, 53)	VAX, Alpha	$.56*10^{-308}$ to $.90*10^{308}$	15 decimal digits
T Float	64 (11, 54)	Alpha, Itanium	$2.23*10^{-308}$ to $1.80*10^{308}$	15 decimal digits

OpenVMS Floating Point Support

	VAX	Alpha	Itanium
Hardware Supported Formats	VAX	VAX IEEE	IEEE
Compiler Default Format		VAX	IEEE
Via Compiler Qualifier		IEEE	VAX (in software)

Rdb 7.2 IEEE FP Support

- Rdb uses a mix F/G and S/T Float internally
- On disk format is F/G float
- Internal FP Computations Use Native Types
  - F/G on Alpha, S/T on Itanium
  - Optimizer computation - costs, comparing strategies
  - REAL, FLOAT, DOUBLE PRECISION computation
    - ☒Computed columns and triggers
    - ☒Declared variables in stored modules & MSPs
    - ☒Computation involving FP columns in stored modules/MSPs
  - Built in statistical functions (AVG, STDDEV, etc.)

Rdb 7.2 IEEE FP Support (Cont.)

```
SQL> create table SALES_BY_SALESMAN (
cont>  SALESMAN_ID char(5),
cont>  SALE_AMOUNT integer(2),
cont>  SALE_DATE date )
cont> ;
SQL> create table SALESMAN_STATS (
cont> SALESMAN_ID char(5),
cont> PERIOD_START date,
cont> PERIOD_END date,
cont> SALE_AVERAGE float,
cont> BONUS computed by (SALE_AVERAGE * 2.0) )
cont> ;
```

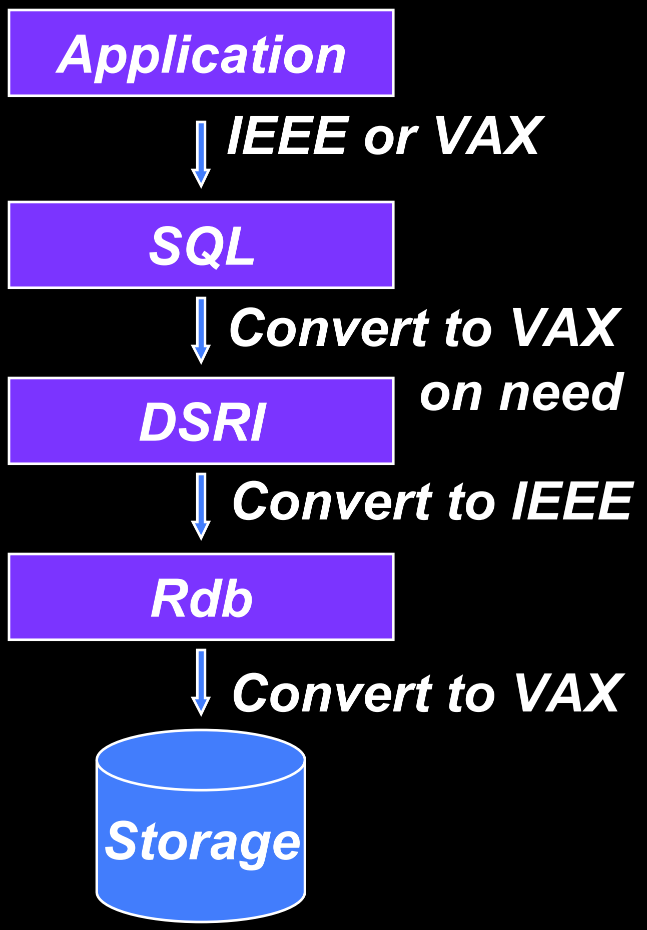
Rdb 7.2 IEEE FP Support (Cont.)

```
SQL> create module PROCESS_SALES language SQL
cont> procedure COMP_SALES_AVE (
cont> :START_DATE date, :END_DATE date );
cont> begin
cont> for :SM as each row of
cont>  read only table cursor SM_CURS for
cont>   select SALESMAN_ID, AVG(SALE_AMOUNT) SMA
cont>   from SALES_BY_SALESMAN
cont>   where (SALE_DATE >= :START_DATE and
cont>   SALE_DATE < :END_DATE) group by SALESMAN_ID
cont> do
cont>   insert into SALESMAN_STATS values (
cont>     :SM.SALESMAN_ID, :START_DATE
cont>     :END_DATE, SM.SMA);
cont> end for; end; end module;
```

Rdb 7.2 IEEE FP Support (Cont.)

- SQL$PRE and SQL$MOD support IEEE interface
  - Supported since Rdb 7.1.0.2
  - Both Alpha and Itanium
  - Key is new /FLOAT qualifier for both preprocessors
    - ☒/FLOAT={D_FLOAT,G_FLOAT,IEEE_FLOAT}
    - ☒Default on Itanium is IEEE_FLOAT
    - ☒/(NO)G_FLOAT retained for backwards compatibility
- DSRI interface doesn’t internally support IEEE types
  - Affects all DSRI clients: SQL, RDO, and precompilers
  - Causes some extra conversions internally
  - Minimal visible effect on applications
  - DSRI will be enhanced for IEEE types in a future release

Rdb 7.2 IEEE FP Support (Cont.)



Application

IEEE or VAX

SQL

Convert to VAX on need

DSRI

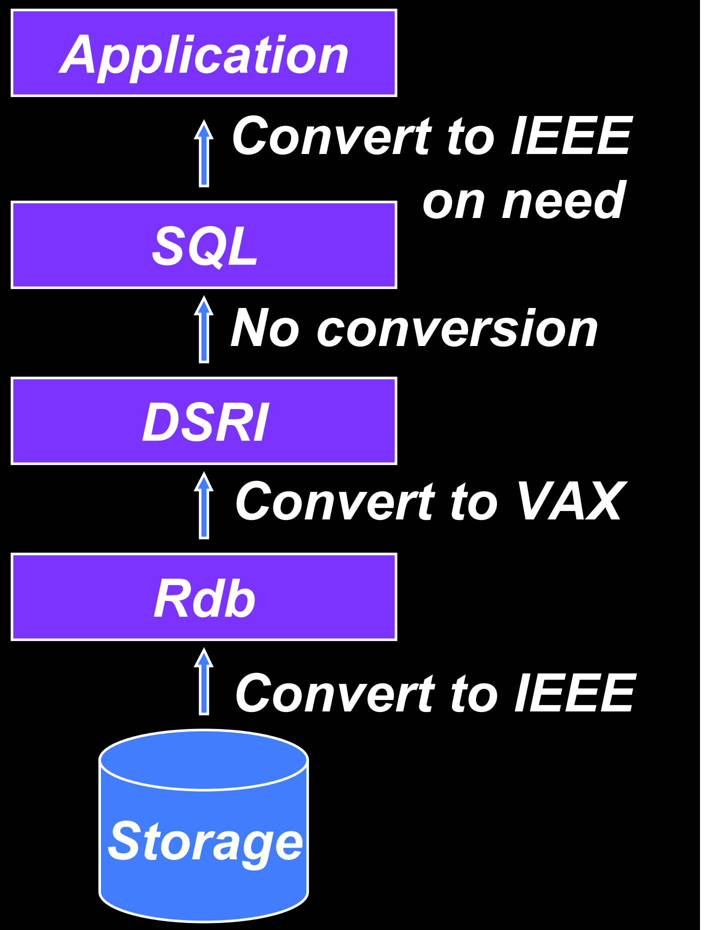
Convert to IEEE

Rdb

Convert to VAX

Storage

Rdb Floating Point Conversions on Itanium



Application

Convert to IEEE on need

SQL

No conversion

DSRI

Convert to VAX

Rdb

Convert to IEEE

Storage

Rdb 7.2 IEEE FP Support (cont.)

- CDD does not support IEEE types
  - Constrains applications using CDD to VAX floats
  - Not likely to change in the near future
- RDML and RDBPRE do not support IEEE types
  - Constrains applications to VAX floats
  - We solicit customer input on whether this is important
- DBMS does not support IEEE types
  - Constrains applications to VAX floats
  - Must compile with /FLOAT=G_FLOAT on Itanium

IEEE FP Features to Note

- Not a number (NaN) values
  - IEEE 754 has what amounts to a built-in NULL, e.g. result of SQRT(-1.0) will be a NaN
  - Use NULL indicators in applications
- Infinity
  - IEEE 754 has an infinity value (really an overflow), e.g. divide by zero will yield a infinity value
  - Useful in some floating point arithmetic
- NaN and Infinity values can not be converted to F/G float
- Avoid both with /IEEE_MODE=FAST

IEEE FP Features to Note (cont.)

- Range differences for VAX/IEEE conversion
  - Both IEEE types have greater range on both ends
  - The “hidden” bit makes this possible
  - Shows up as underflow and overflow on conversion to F or G Float
  - Precision differences can also cause accuracy loss if you convert from IEEE to VAX and back again.
- Round Even
  - VAX FP arithmetic rounds up – IEEE rounds even
  - Differences will inevitability be seen on Itanium because all arithmetic is IEEE 754

Round Up vs. Even Example

```
$ TYPE X.BAS
 declare decimal(7,3) pp
 declare integer pn

 pn = 116
 pp = pn/10
 print "pp1 = "; pp

 pn = 331
 pp = pn/10
 print "pp2 = "; pp
```

This is a real example from our regression test system

Round Up vs. Even Example (Cont.)

VAX

```
$ BASIC X
$ LINK X
$ RUN X
pp1 = 11.6
pp2 = 33.099
```

Alpha (VAX float)

```
$ BASIC X /REAL=SINGLE
$ LINK X
$ RUN X
pp1 = 11.6
pp2 = 33.099
```

Alpha (IEEE float)

```
$ BASIC X /REAL=SFLOAT
$ LINK X
$ RUN X
pp1 = 11.599
pp2 = 33.1
```

Itanium

```
$ BASIC X /REAL={any}
$ LINK X
$ RUN X
pp1 = 11.599
pp2 = 33.1
```

General Application Porting Tips

- Compile with /FLOAT qualifier everywhere
  - Ensures consistent format for parameters
  - Watch out if your language has types with explicit floating point format
  - Get rid of obsolete /(NO)G_FLOAT qualifiers
- Defaults are different between architectures
  - Applies to both SQL$MOD and SQL$PRE
  - Matches compiler default on Alpha, IEEE on Itanium
- Suppress IEEE special values
  - /IEEE_MODE=FAST with some languages
  - /FAST or /MATH_LIBRARY=FAST with others

General Application Porting Tips (Cont.)

- Watch out for other shared libraries
  - Any other shared libraries must use consistent floating point types
  - HP math libraries on OpenVMS 7.3-1 and earlier on Alpha take F/G Float parameters
  - OpenVMS 7.3-2 and later also have math libraries with format-specific entry points (e.g. square root has MTH$GSQRT, MTH$SSQRT, and MTH$TSQRT in lieu of just MTH$SQRT)
  - See the HP OpenVMS documentation for more
- You can try out IEEE floating point today on Alpha with Rdb 7.1.0.2 or later

BASIC Porting Tips

- No SQL$PRE with BASIC (not a change)
- BASIC has some explicitly formatted floating point types: GFLOAT, SFLOAT, and TFLOAT
- It also has generics: REAL, SINGLE, DOUBLE
  - Default floating point format varies by architecture
  - There is /REAL_SIZE qualifier that lets you specify the floating point format, e.g. GFLOAT, SFLOAT, TFLOAT
  - Stay away from the /SINGLE and /DOUBLE qualifiers
- Recommend using “REAL” type and /REAL_SIZE qualifier to make application portable

C Porting Tips

- Floating point types
  - C has no explicitly formatted floating point types
  - Type “float” is 32 bits, “double” is 64 bits
  - Default format varies by architecture
- CC /FLOAT qualifier matches SQL$MOD and SQL$PRE /FLOAT exactly
- Use /IEEE_MODE=FAST to preserve application behavior

ANSI C 99

- ANSI C allows specification of NAN and INFINITY literal values
- These values are not support for insert into Oracle Rdb
- Requires use a <math.h> and <fp.h> include libraries
- Use isnan() and isfinite() builtins and set the NULL indicator before insert

ANSI C Example

```
#include <fp.h>
…
double ieee_data;
…
    ieee_data = INFINITY;
    printf("data=%lf\n", ieee_data);
    execute_stmt (&sqlca, &stmtid, sqlda );
…
    ieee_data = NAN;
    printf("data=%lf\n", ieee_data);
    execute_stmt (&sqlca, &stmtid, sqlda );
```

ANSI C Example…

```
$ RUN IEEETEST
data=Infinity
Error -304 encountered in main:INSERT
-%RDB-E-ARITH_EXCEPT, truncation of a numeric value
at runtime
-COSI-F-INVCVT, invalid data type conversion
insert failed
data=NaNQ
Error -304 encountered in main:INSERT
-%RDB-E-ARITH_EXCEPT, truncation of a numeric value
at runtime
-COSI-F-BADPARAM, bad parameter value
insert failed
```

COBOL Porting Tips

- Floating point types
  - COBOL has no explicitly formatted floating point types
  - Type “COMP-1” is 32 bits, “COMP-2” is 64 bits
  - Default format varies by architecture
- COBOL /FLOAT qualifier matches SQL$MOD and SQL$PRE /FLOAT exactly
- No /IEEE_MODE or equivalent qualifier
  - Current behavior is like FAST
  - Documentation is silent about IEEE 754

FORTRAN Porting Tips

- Floating point types
  - Fortran has no explicitly formatted floating point types
  - Types “real” and “real*4” are 32 bits
  - Types “double precision” and “real*8” are 64 bits
  - Default format varies by architecture
- C /FLOAT qualifier matches SQL$MOD and SQL$PRE /FLOAT exactly
- Use /IEEE_MODE=FAST to preserve application behavior
  - This is the default
  - /MATH_LIBRARY=FAST or /FAST overrides /IEEE

Pascal Porting Tips

- Pascal has some explicitly formatted floating point types: F_FLOAT, D_FLOAT, G_FLOAT, S_FLOAT, and T_FLOAT
- It also has generic types:
  - REAL and SINGLE are 32 bit, DOUBLE is 64 bit
  - Default floating point format varies by architecture
  - /FLOAT qualifier that lets you specify the format
  - The Pascal FLOAT attribute also allows format specification inside the program
- No /IEEE_MODE qualifier

For More Information

- Rdb 7.1 SQL Reference Manual
- http://cch.loria.fr/documentation/IEEE754/
- http://cch.loria.fr/documentation/IEEE754/ACM/goldberg.pdf and here
- http://www.hp.com/products1/evolution/alpha_retaintrust/download/i64-floating-pt-wp.pdf and here
- www.oracle.com/rdb
- metalink.oracle.com
- www.hp.com/products/openvms
- john.howard@oracle.com

QUESTIONS

ANSWERS

ORACLE®