Subject: Decimal
Posted by mirek on Fri, 01 Aug 2008 08:10:31 GMT
View Forum Message <> Reply to Message

Well, it looks like Firebird seems to force us to add Decimal support.

I was googling a bit and found there two relevant topics:

http://docs.python.org/lib/module-decimal.html
http://www.yoda.arachsys.com/csharp/decimal.html

It looks like there are some standards -> I would recommend to follow them here.

Mirek

Subject: Re: Decimal
Posted by masu on Fri, 01 Aug 2008 08:31:54 GMT
View Forum Message <> Reply to Message

There is also a WP article on DFP:

http://en.wikipedia.org/wiki/Decimal_floating_point

It also contains links to the actual IEEE standard.

Matthias

Subject: Re: Decimal
Posted by mirek on Fri, 01 Aug 2008 09:18:37 GMT
View Forum Message <> Reply to Message

Well, completely opposite idea...

Maybe we could use int128 (supported by both MSC and GCC) and sort of fixed point, only removing last 4 or 5 bits of number to specify the precission (as number of significant places).

That would give as ability to represent 10^20 numbers with up to 10^-16 precision (if I have computed numbers well .

Mirek

Subject: Re: Decimal

Posted by aldeacity on Sat, 02 Aug 2008 11:30:02 GMT
View Forum Message <> Reply to Message

Hello,
perhaps int128 would be a very good idea . But, sorry if I make a mistake, but, until now, how does Oracle for Upp manage number(x,y) data?

Juan.

Subject: Re: Decimal
Posted by unodgs on Sat, 02 Aug 2008 11:54:32 GMT
View Forum Message <> Reply to Message

aldeacity wrote on Sat, 02 August 2008 07:30Hello,
perhaps int128 would be a very good idea . But, sorry if I make a mistake, but, until now, how does Oracle for Upp manage number(x,y) data?

Juan.
All decimal types in other databases are converted to double. I will try to implement int128 based decimal.

Subject: Re: Decimal
Posted by mirek on Sat, 02 Aug 2008 12:21:07 GMT
View Forum Message <> Reply to Message

Well, here is something to start with

```
#include <Core/Core.h>

using namespace Upp;

double pow2(int x)
{
 double b = 1;
 while(x--)
  b *= 2;
 return b;
}

typedef __int128_t int128;

#define I128_10_17 (int128)I64(1000000000) * (int128)I64(100000000)

struct Decimal {
 int128 data;
```

```
 static int128 e10[38];

public:
 void Add(const Decimal& b) {
  byte sa = data & 15;
  byte sb = b.data & 15;
  data = (~(int128)15 & data) + (~(int128)15 & b.data) | max(sa, sb);
 }
 void Mul(const Decimal& b) {
  byte sa = data & 15;
  byte sb = b.data & 15;
  data = (((~(int128)15 & data) * (~(int128)15 & b.data)) / I128_10_17) | max(sa + sb, 16);
 }

 String ToString() const;
 const char *Scan(const char *s);

 Decimal();
};

int128 Decimal::e10[38];

Decimal::Decimal()
{
 static bool init;
 if(!init) {
  int128 q = 1;
  for(int i = 0; i < 38; i++) {
   e10[i] = q;
   q = q * 10;
  }
 }
 data = 0;
}

String Format128(int128 a)
{
 RTIMING("Format128");

 if(a < 0)
  a = -a;
 char b[50];
 char *p = b + 50;
 do {
  *--p = a % 10 + '0';
  a = a / 10;
 }
```

```
 while(a);
 return String(p, b + 50);
}

const char *Decimal::Scan(const char *s)
{
 while(*s == ' ')
  s++;
 bool neg = false;
 if(*s == '-') {
  neg = true;
  s++;
 }
 while(*s == ' ')
  s++;
 if(!IsDigit(*s))
  return NULL;
 data = 0;
 while(IsDigit(*s))
  data = *s++ - '0' + 10 * data;
 int digits = 0;
 if(*s == '.') {
  s++;
  while(IsDigit(*s) && digits < 16) {
   data = *s++ - '0' + 10 * data;
   digits++;
  }
  while(IsDigit(*s)) s++;
 }
 data *= e10[16 - digits];
 if(neg)
  data = -data;
 data = (data & ~(int64)15 & data) | digits;
 return s;
}

String Decimal::ToString() const
{
 int digits = data & 15;
 String x = Format128(~(int64)15 & data);
 int n = x.GetLength();
 if(n <= 16)
  return "0." + x.Mid(0, digits);
 return x.Mid(0, n - 16) + "." + x.Mid(n - 16, digits);
}

Decimal Dec(const char *s)
{
```

```
 Decimal d;
 d.Scan(s);
 return d;
}

inline Decimal operator+(const Decimal& a, const Decimal& b)
{
 Decimal r = a;
 r.Add(b);
 return r;
}

inline Decimal operator*(const Decimal& a, const Decimal& b)
{
 Decimal r = a;
 r.Mul(b);
 return r;
}

CONSOLE_APP_MAIN
{
 DUMP(Dec("1.20"));
 DUMP(Dec("1.20") + Dec("2"));
 DUMP(Dec("1.02") + Dec("1.005"));
 DUMP(Dec("1.02") * Dec("1"));
 DUMP(Dec("1.02") * Dec("0.50"));
 DUMP(Dec("1.02") * Dec("1.005"));
 return;

 RDUMP(Format128(I128_10_17));

 int64 x = I64(0x1234123412341234);
 DUMP(Format64Hex(x & ~(int64)15));
 Decimal a;
 Decimal b;
 a.Add(b);
 for(int i = 0; i < 128; i++) {
  LOG(i << ' ' << pow2(i));
 }
}
```

Conversions and + work, * not yet... You will need to adjust typedef for MSC.

Mirek