The Third Manifesto
Why? What? How?

Hugh Darwen
hd@thethirdmanifesto.com
www.thethirdmanifesto.com

Ref: *Databases, Types, and The Relational Model* (2006)
(soon to appear at the web site as free PDF download)

for Palacky University, Olomouc

May 8th, 2014

Why *TTM*?

1989-1990: Publication of two “manifestos”:
First, *The Object-Oriented Database Systems Manifesto,*
then, in response
Third-Generation Database System Manifesto
concerning future database systems.

HD and Chris Date didn’t agree with either of them, so decided to produce another. *TTM* first appeared in 1994.

Why didn’t they agree with either of them?

Why Not OODB?

Object Orientation is for software engineering, not for a database that is available to a wide and diverse community, supporting ad hoc query and integrity via declared constraints.

Codd's relational proposal (1970) raised the level of abstraction. OODB lowers it again (as do NoSQL and Big Data now).

The second manifesto agreed with that objection but equated relational to SQL.

(In any case, nothing much came of OODB)

Why Not SQL?

SQL fails to adhere to the relational model of data.

SQL's deviations introduce needless complications and traps for the unwary.

SQL is a very badly designed language. Many generally accepted principles of good language design are violated.

But why did HD and CJD care enough to write *TTM*?

Events in HD's Life
leading to *TTM*

1967 : Joined an IBM "Service Bureau" in England

1969 : "Terminal Business System" – putting users in direct contact with their databases

1972 : Attended Date's course on database (a personal watershed)

1978 : "Business System 12" - a relational dbms for the Bureau Service

1985 : Death of Bureau Service (and of BS12)

1987 : Attended a database conference at which Ted Codd and Chris Date both spoke against SQL.
Joined group developing international standard for SQL.
Collaboration with Date started

1994 : Event at SQL standards meeting, Munich, causes HD to make first draft of *TTM* (in disgust!).

What's Wrong with SQL?

Relational Model Violations:

- A table column can have no name
- Two columns in same table can have the same name …
- … so columns are ordered
- The same row can appear more than once in a table
- NULL and 3-valued logic
- Incomplete support for constraints
- Relations of degree zero not supported
- "=" operator gives *true* for some unequal character strings ...
- … resulting in indeterminacy for some operations
- 1999: Introduction of pointers ("REF values")
- 1999: CREATE TABLE *tablename* OF TYPE *typename*

1999 violations arose from mistaken thinking in OO community

What's Wrong with SQL?

Violations of Good Language Design Principles:

- Rigid query structure not easily extensible
- Lack of orthogonality (much "ad hoc-ery" instead) …
- … e.g. 2011 extensions for temporal support
- Not all types are "first class"
- Placing an expression in parens changing its meaning
- Coercion (implicit type conversion according to context)
- [and lots more]

See

- *The Askew Wall* (at www.dcs.warwick.ac.uk/~hugh/TTM/presentations
- *SQL: A Comparative Survey* by HD (free download from Bookboon.com)

The Perversity of SQL

```
SELECT CityName
FROM City C1
WHERE 4 > (SELECT COUNT(*)
           FROM City C2
           WHERE C1.Population < C2. Population)
```

The Unperversified Version

```
SELECT CityName
FROM City C1
WHERE  (SELECT COUNT(*)
        FROM City C2
        WHERE C2. Population > C1. Population) < 4
```

In Tutorial D

```
WITH { City_C1 := City RENAME { PREFIX ‘’ AS ‘C1_’ },
       City_C2 := City RENAME { PREFIX ‘’ AS ‘C2_’ } :

( City_C1 WHERE COUNT( City_C2 WHERE
                       C2_Population > C1_Population) < 4 )
{ C1_CityName}
```

- No restriction on order of comparands
- No coercion — aggregate operator COUNT is defined to take a relation and return an integer.
- No SQL-style dot qualifiers in attribute references

What is *TTM*?

A prescription for correct design of a relational database language

Numbered lists of:

- RM Prescriptions (RM = Relational Model)
- OO Prescriptions (OO = “Other Orthogonal” ☺)
- RM Proscriptions
- OO Proscriptions
- RM Very Strong Suggestions
- OO Very Strong Suggestions

Proscriptions, inspired by errors in SQL, are implied by Prescriptions

Also, we address certain problems and confusions arising from E.F. Codd’s papers

E.g. RM Prescriptions 26 and 8

26. **D** shall be constructed according to well established principles of **good language design**.

Note use of "**D**" as a generic language name.

The other 25 are much less informal and rather more precise, e.g.:

8. **D** shall support the equality comparison operator "=" for every type *T*. Let *v1* and *v2* be values, and consider the equality comparison *v1* = *v2*. The values *v1* and *v2* shall be of the same type *T*. The comparison shall return TRUE if and only if *v1* and *v2* are the very same value.

(Some critics have urged for *greater* formality.)

Problems Caused by Codd's Papers

- Domain? Was this the same as type? Definition seemed to imply that but he said not.
- Relation attributes: not ordered, said Codd, but he said little about implications — e.g., requirement for unique names
- Relational algebra incomplete (no extension, no aggregation, no attribute renaming) and some operators unclear (e.g., UNION)
- Relational calculus: Notation Alpha found unsound on close scrutiny
- First Normal Form: imprecise definition ("atomic"?) and unclear intention — did he mean it for all relations, or was it just a db design principle, like the other NFs?
- Post 1970: NULL and 3VL (proposal later withdrawn in favour of something even worse: two kinds of null and 4VL)

So, what does *TTM* do about these problems?

TTM Clarifications of Codd's RM

- Domain: Term replaced by type and RM Pres 1-8 specify in detail the required properties for "scalar" types, tuple types, relation types

- Relation attributes—RM Pre 9:

  9. A **heading** {*H*} is a set of ordered pairs or **attributes** of the form <*A,T*>, where:

     a. *A* is the name of an **attribute** of {*H*}. No two distinct pairs in {*H*} shall have the same attribute name.

- Relational algebra—RM Pre 18:

  18. **D** shall support the usual operators of the **relational algebra** (or some logical equivalent thereof).

- First Normal Form: An attribute can be of any type, including tuple types and relation types.

Some Notable RM Proscriptions

3. **D** shall include no concept of a "relation" containing two distinct tuples *t1* and *t2* such that the comparison "*t1* = *t2*" evaluates to TRUE. It follows that … for every relation *r* expressible in **D**, the tuples of *r* shall be distinguishable by value.

So SQL's "duplicate row" phenomenon is outlawed.

4. **D** shall include no concept of a "relation" in which some "tuple" includes some "attribute" that does not have a value.

SQL NULL is not a value because it violates *TTM*'s definition of "="

5. **D** shall not forget that relations with no attributes are respectable and interesting, …

RM Pre 9 doesn't specify "nonempty" for the set of attributes constituting a heading.

OO Proscriptions

1. Relvars [relation variables] are not domains. [domain = type]

   So SQL's CREATE TABLE *tablename* OF TYPE *typename* is outlawed

2. No database relvar shall include an attribute of type *pointer.*

   So oids and SQL's REF values are outlawed

Summary of Features Required by *TTM*

- Types: Strong typing, static type checking, all types first class, user-defined types (optional extra: subtyping by constraint)

- Operators: support for user-defined functions and procedures, including relational operators

- Variables: of any type, but only relation variables in database
  - "multiple assignment" for simultaneous update of several
  - assignment operator, e.g. ":=", for all variables
  - relvars can be real (base) or virtual (view)

- Comparisons: "=" for all types, including user-defined and relations
  "⊆", "⊂", etc. for comparing relations, guaranteeing complete support for integrity constraints

- Completeness: computational, relational, integrity constraints

- Transactions: can be synchronously nested

So that's the "Why" and the "What". Now, what about the "How"? …

TTM in Action

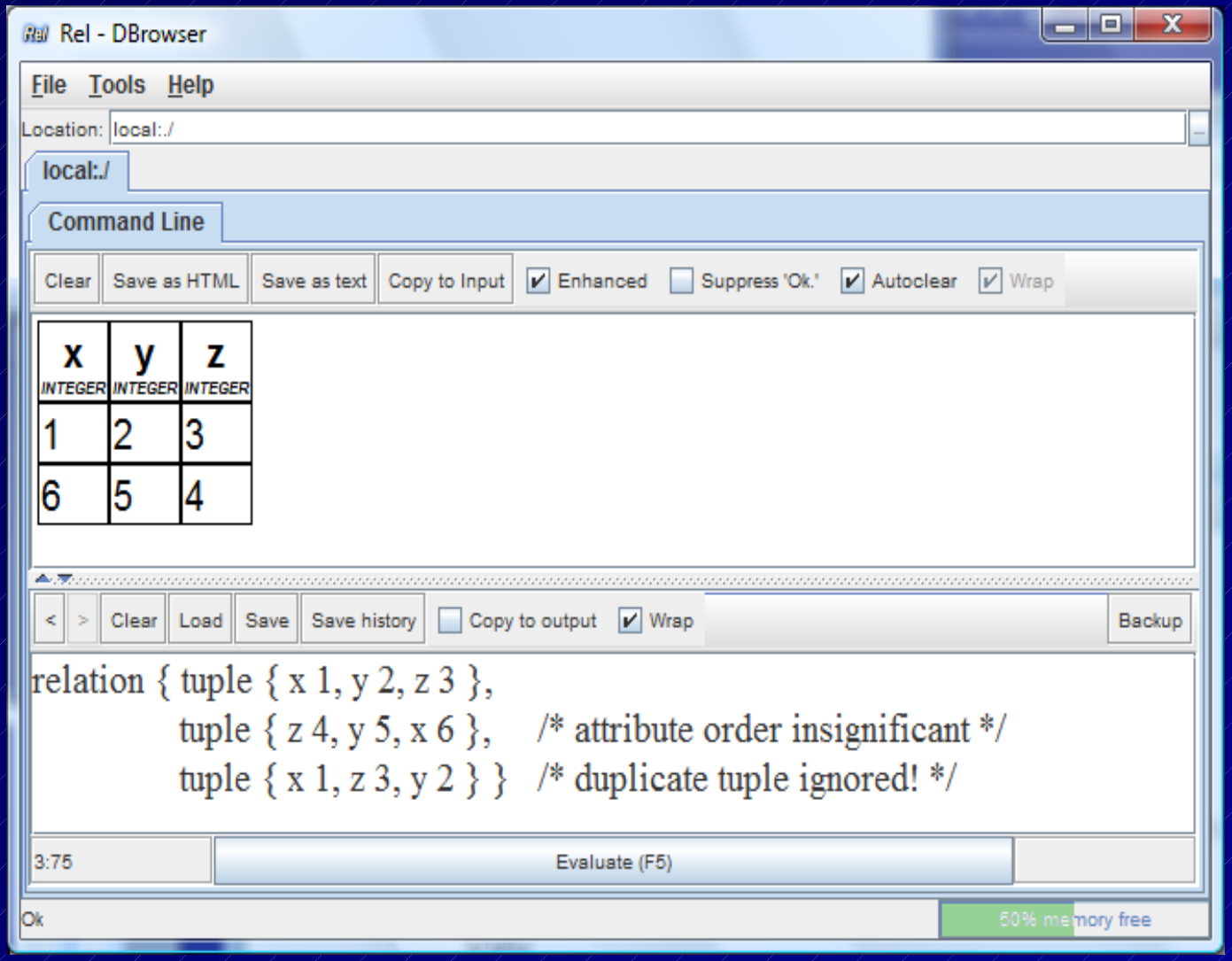
Some of HD's experiences with *Rel*, an implementation of **Tutorial D** by Dave Voorhis (University of Derby, UK)

(Suggesting title *The Relational Model of Data for Small, Private Data Banks* for a possible paper ☺)

First, an illustration of relational correctness …

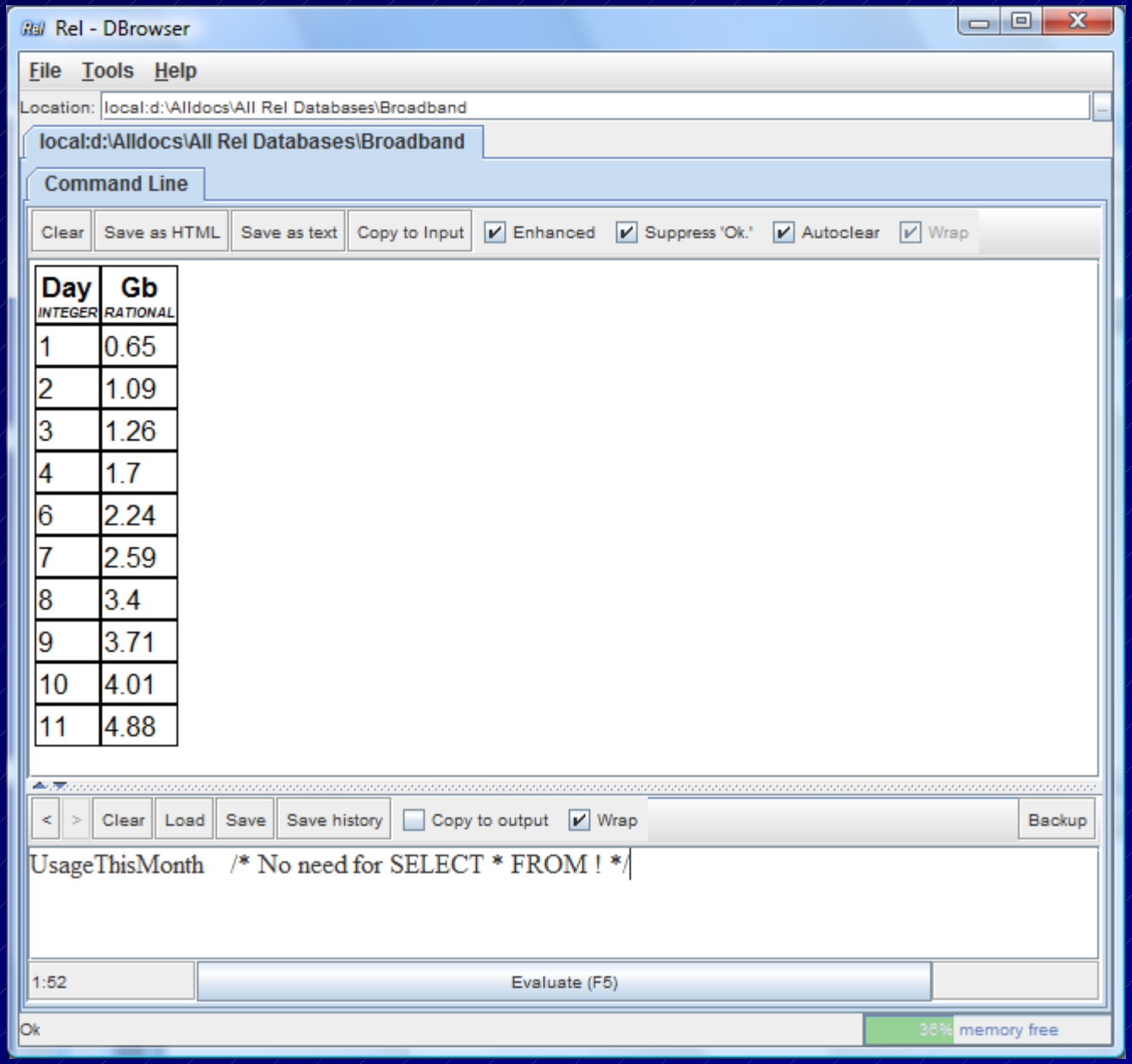
Relational Correctness

Evaluation of a relation literal:



No real counterpart in SQL: VALUES is order dependent and not available everywhere.

Obtain Current Value of a Base Relvar

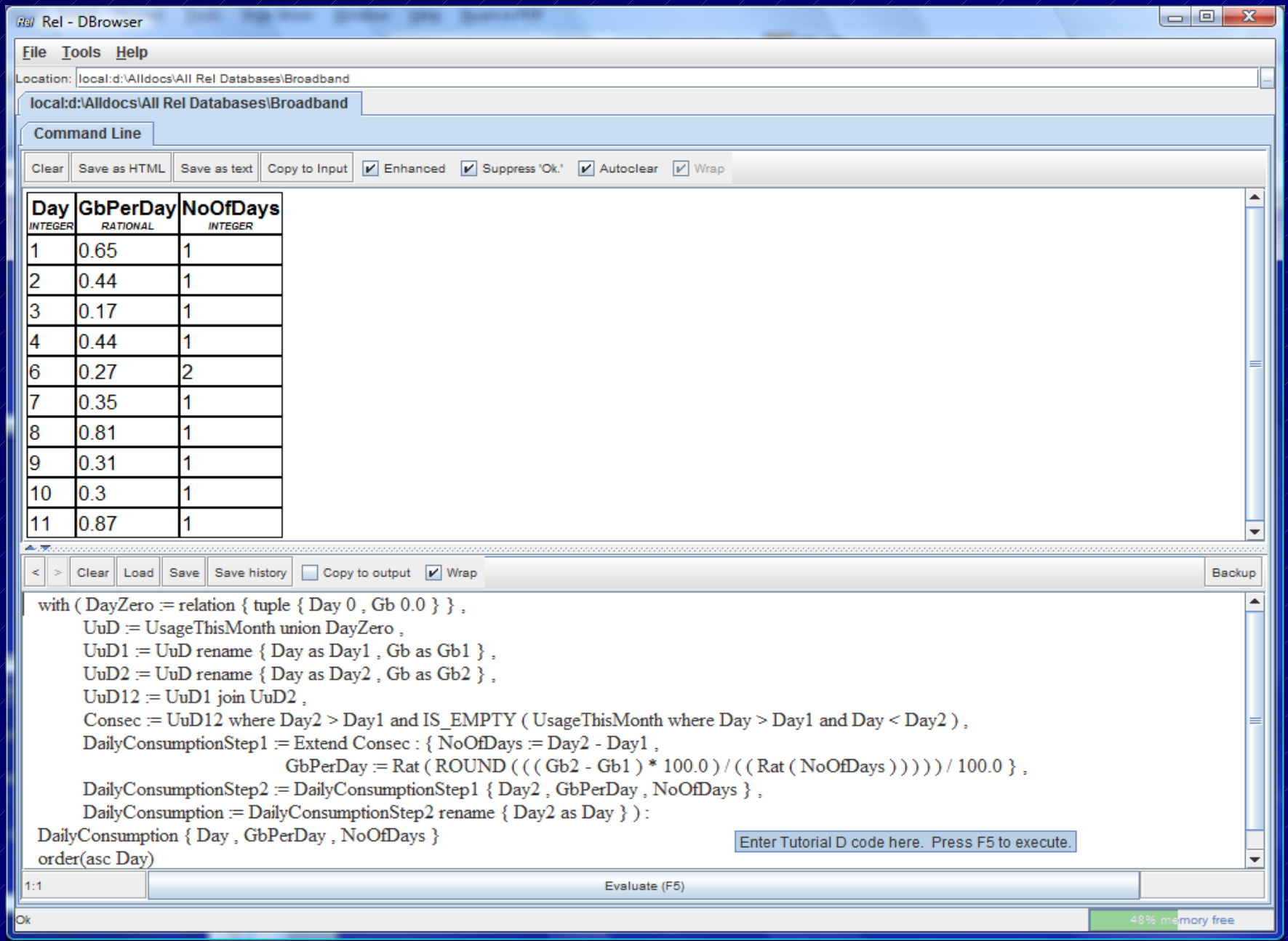


Daily readings of broadband consumption.

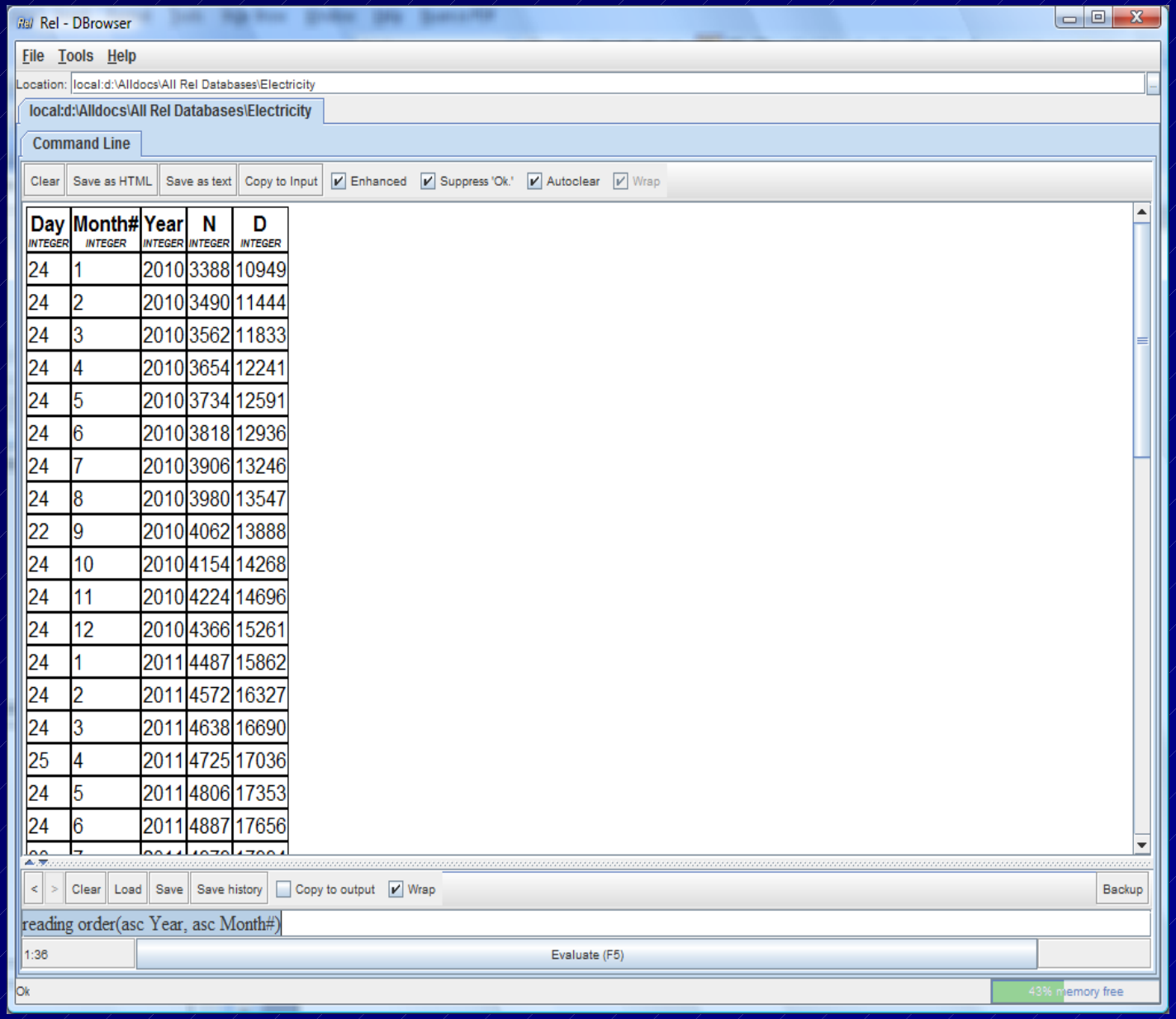
Note the gap, no reading on day 5.

Is this easier than SQL?

A Difficult Query: Daily Consumption



Domestic Electricity Meter Readings

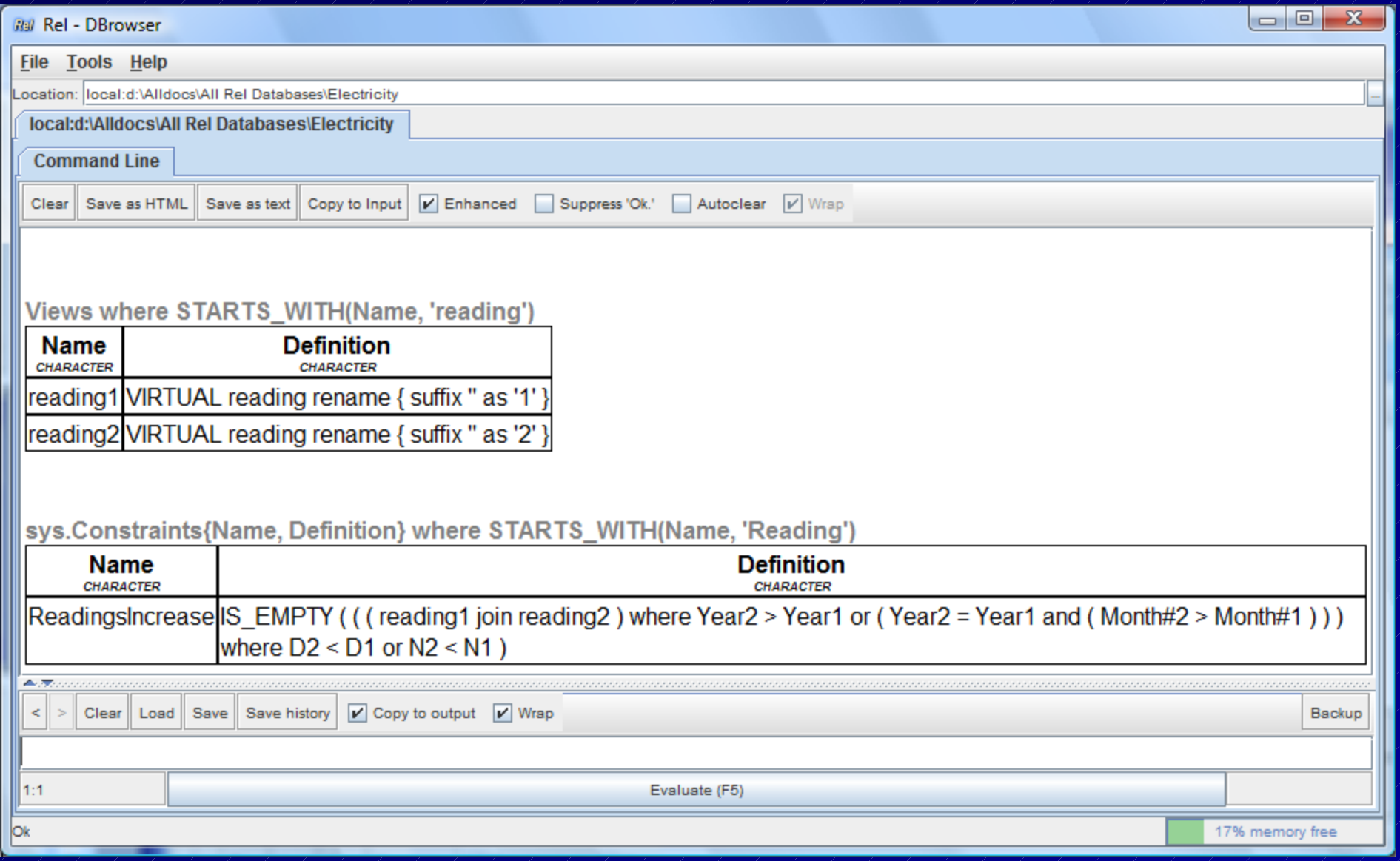


Day INTEGER	Month# INTEGER	Year INTEGER	N INTEGER	D INTEGER
24	1	2010	3388	10949
24	2	2010	3490	11444
24	3	2010	3562	11833
24	4	2010	3654	12241
24	5	2010	3734	12591
24	6	2010	3818	12936
24	7	2010	3906	13246
24	8	2010	3980	13547
22	9	2010	4062	13888
24	10	2010	4154	14268
24	11	2010	4224	14696
24	12	2010	4366	15261
24	1	2011	4487	15862
24	2	2011	4572	16327
24	3	2011	4638	16690
25	4	2011	4725	17036
24	5	2011	4806	17353
24	6	2011	4887	17656

N = night, D = day

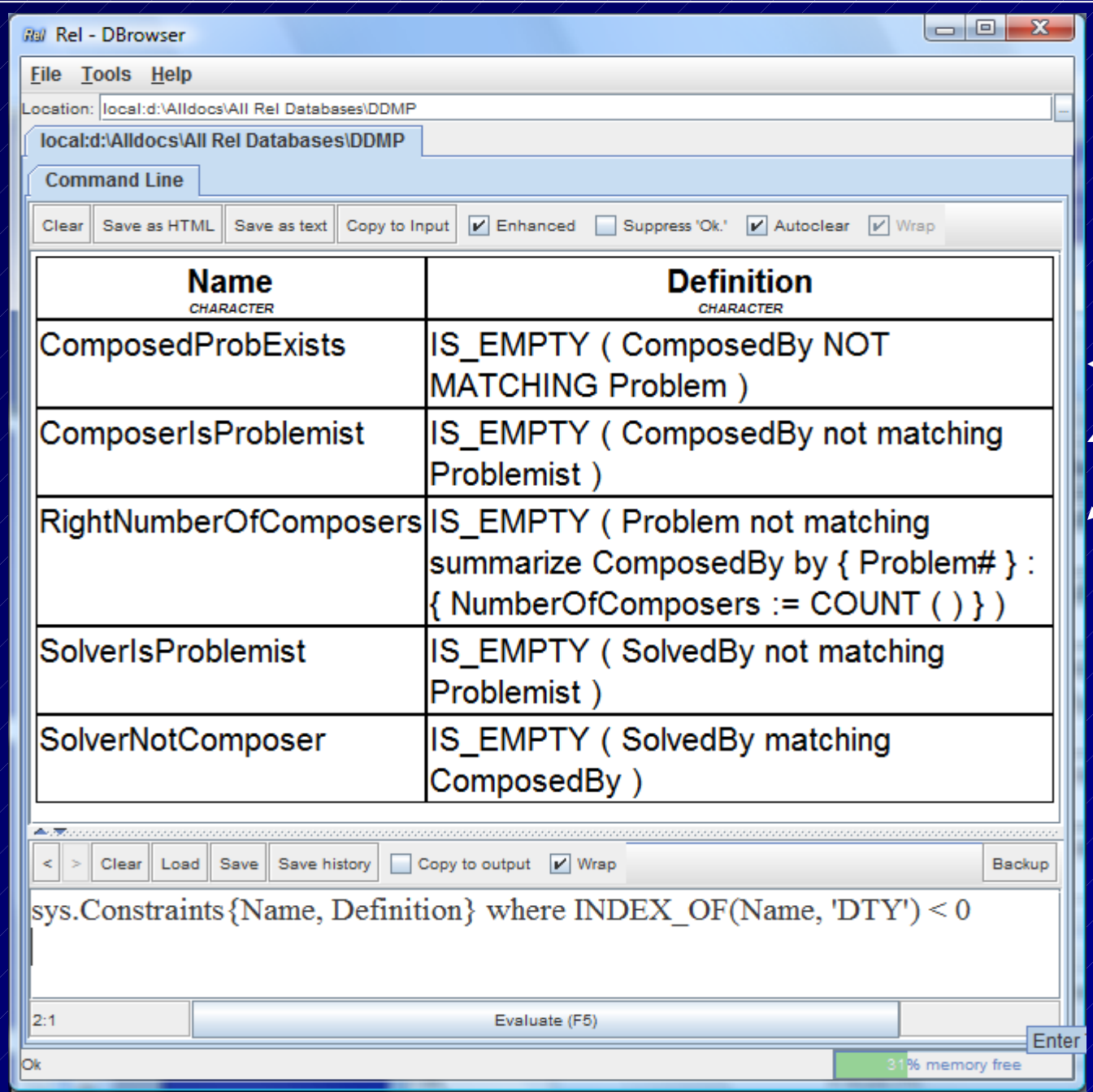
I once added an incorrect reading (much too large), causing (a) some alarm that month, and (b) addition of the constraint shown on the next slide …

A Very useful Constraint



I.e., readings must ascend. Note use of suffix-renaming.

Constraints for Bridge Problems Database

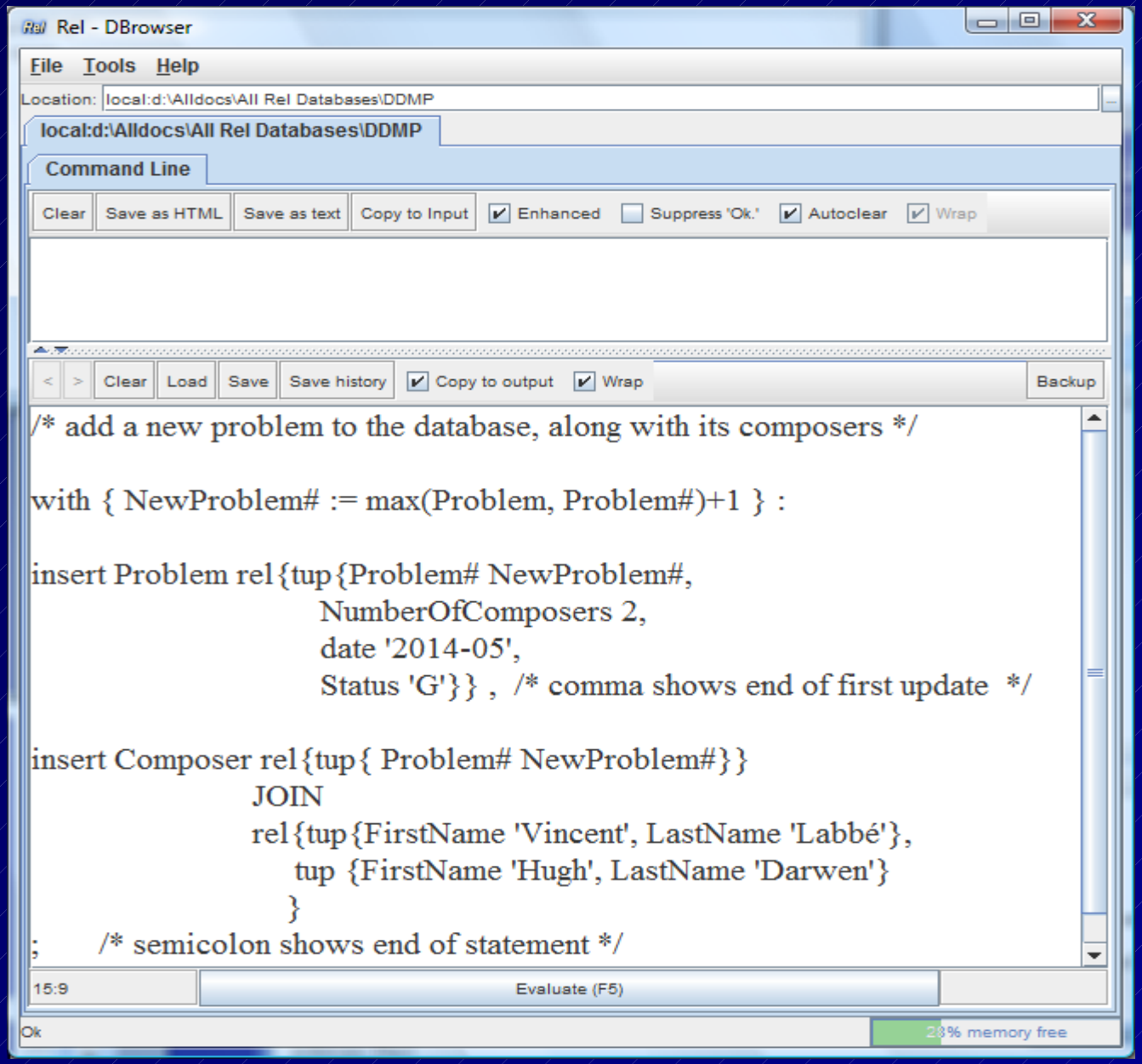


Generalisation of "foreign key" constraints

Can't be done in any(?) SQL products.

Nor can this one.

Multiple Assignment



I cheated a little here: WITH for statements isn't in *Rel* yet.

Constraints checked at the semicolon, not the comma.

Note neat trick to avoid repetition of Problem#.

The Relationlander's Promise

I solemnly promise …

… *never* to use the word "relational" when I mean SQL, …

… cross my heart and hope to die.

The End