Copyright ©2005 Timothy Howard Merrett Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation in a prominent place. Copyright for components of this work owned by others than T. H. Merrett must be honoured. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to republish from: T. H. Merrett, School of Computer Science, McGill University, fax 514 398 3883.

The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding agencies.

T. H. Merrett

Trie Joins

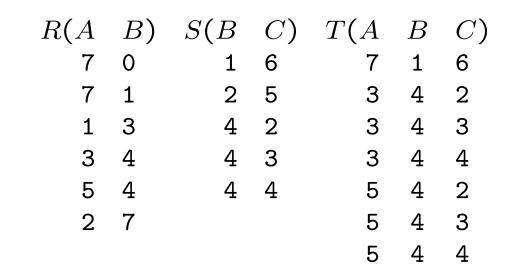
T. H. Merrett, McGill University

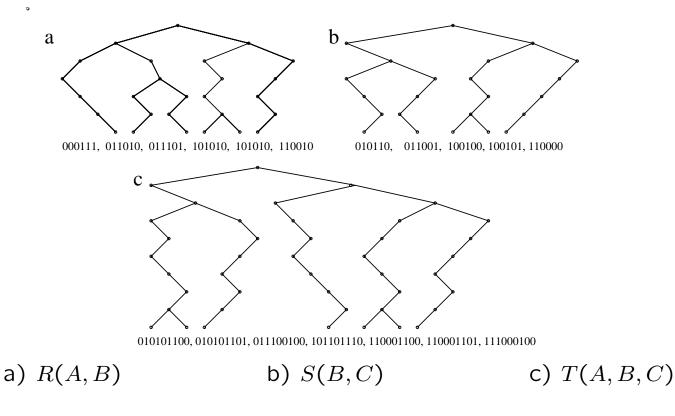
Objective: represent two relations as kd-tries and compute directly the kd-trie representing their natural join.

Benefit: work purely with tries, without decompressing the data.

T. H. Merrett

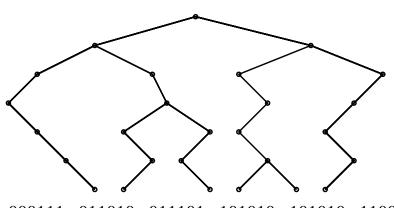
Three relations as kd-tries





T. H. Merrett

R(A,B) as bitpairs



000111, 011010, 011101, 101010, 101010, 11001(

11

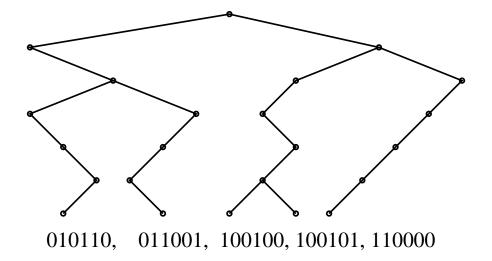
11 11

c

- 10 01 01 10
- 01 11 10 10
- 01 01 10 01 01
- 01 10 01 11 10

T. H. Merrett

S(B,C) as bitpairs

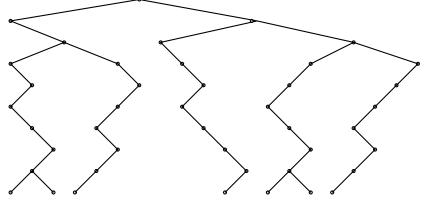


11

- 01 11
- 11 10 10
- 01 10 01 10
- 01 10 10 10
- 10 01 11 10

T. H. Merrett

T(A, B, C) as bitpairs



010101100, 010101101, 011100100, 101101110, 110001100, 110001101, 1110001

- 11
- 01 11
- 11 01 11
- 01 01 01 10 10
- 10 10 10 10 10
- 01 10 01 01 10
- 01 01 01 01 01
- 10 10 01 10 10
- 11 10 10 11 10

T. H. Merrett

T < -R ijoin S

	R(A,B)	S(B,C)	T(A, B, C)					
	11		11					
J	11 11	11	01 11					
		01 11	11 01 11					
	10 01 01 10		01 01 01 10 10					
J	01 11 10 10	11 10 10	10 10 10 10 10					
		01 10 01 10	01 10 01 01 10					
	01 01 10 01 01		01 01 01 01 01					
J	01 10 01 11 10	01 10 10 10	10 10 01 10 10					
		10 01 11 10	11 10 10 11 10					

Note that the rows cycle:

• left (R)

- both (R, S; join attribute)
- right (S)

The algorithm

- 1. Use the paths in the result so far to predict all possible next steps.
- See whether and whence these come from the source(s): left, both, right.

The first cycle

			righ	t			resu	lt			final
pos	path	bp	lev	pos	path	bp	lev	pos	path	bp	
0		11					0	0		11	
1		11	0	0		11	1	1		11	01
2	r	11						2	r	11	
			1	1	1	01	2	3	11	01	Х
				2	r	11		4	lr	11	
								5	rl	01	
								5	rr	11	
	0 1 2	pos path 0 1 I	pos path bp 0 11 1 I 11 2 r 11	pos path bp lev 0 11 1 l 11 0 2 r 11 1	pos path bp right 0 11 lev pos 1 1 11 0 0 2 r 11 1 1 1 1 1 2 r 11 2	right pos path bp lev pos path 0 11 1 l 11 0 0 2 r 11 1 l 1 l 2 r 2 r	pos path bp right 0 11 lev pos path bp 1 1 11 0 0 11 2 r 11 1 0 0 1 1 1 1 01 2 r 11 1 1	pos path bp right resule 0 11 lev pos path bp lev 1 1 11 0 0 11 1 2 r 11 1 0 0 11 1 2 r 11 1 0 0 2 2 r 11 1 1 0 2	pos path bp right result 0 11 lev pos path bp lev pos 0 11 1 0 0 11 1 1 1 1 11 0 0 11 1 1 2 r 11 1 1 1 2 3 - - 2 1 1 4 2 r 11 1 01 2 3 - 2 r 11 5 5	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$

left

• 11 from left \longrightarrow result.

both

- (1) Result level 1 will have I node and r node.
- (2) Left level 1 has 2 nodes; right level 1 has 1 node: and the Cartesian product, giving result 11, 11.
- (The I node 11 will eventually be corrected to 01 but we don't know this yet.)

right

- (1) Result level 2 will have 4 nodes: II, Ir, rl rr. (II removed: later.)
- (2) The II result must come from left I, both I, so copy over right I: 01.
- (2) The Ir result must come from left I, both r, so copy over right r: 11.
- (2) The rl result must come from left r, both l, so copy over right l: 01.
- (2) The rr result must come from left r, both r, so copy over right r: 11.

T. H. Merrett

The second cycle												
left			-	right	t			resu	lt			final
lev 2	pos 3 4 5 6	path II Ir rI rr	bp 10 01 01 10	lev	pos	path	bp	lev 3	pos 7 8 9 10 11 12	path IIr Irl Irr rr rlr rrl rrr	bp 10 01 01 01 10 10	X
3	7 8 9 10	lll Irr rlr rrl	01 11 10 10	2	3 4 5	lr rl rr	11 10 10	4	13 14 15 16 17 18	llri Irir Irrr rirr rrii rrii	10 01 10 10 10 10 10	×
				3	6 7 8 9	lrl Irr rll rrr	01 10 01 10	5	19 20 21 22 23 24	llrir Iriri Irrri rirri rrili rrrii	10 01 10 01 01 10	×

left

• (1) Result level 3 will have 6 nodes: IIr, Irl, Irr, rlr, rrl, rrr.

.

 (2) The IIr result must come from left I, both I, right r so copy over 10 (the left II node). And so on: both Irl and Irr from left Ir (01), rIr from left rI (01), and both rrl and rrr from left rr (10).

both

- (1) Result level 4 will have 6 nodes: IIrl, Irlr, Irrr, rlrr, rrll, rrrl.
- (2) The IIrl result must come from left I, both I, right r and left I, so and left III (01) with right Ir (11) giving 01.
 Similarly, Irlr comes from left Irr and right rl, etc.

T. H. Merrett

The third cycle (last for this ex.)												
left				right				resu	final			
lev	pos	path	bp	lev	pos	path	bp	lev	pos	path	bp	
4	11	IIIr	01					6	25	llrlrl	01	Х
	12	Irrl	01						26	IrIrIr	01	
	13	Irrr	10						27	Irrrll	01	
	14	rIrl	01						28	rlrrlr	01	
	15	rrll	01						29	rrIIIr	01	
									30	rrrIII	01	
5	16	IIIrr	01	4	10	IrIr	01	7	31	llrlrlr		Х
	17	Irrlr	10		11	Irrl	10		31	Irlrlrr	10	
	18	Irrrl	01		12	rllr	10		32	Irrrllr	10	
	19	rlrlr	11		13	rIII	10		33	rlrrlrr	01	
	20	rrllr	10						34	rrIIIrr	10	
									35	rrrIIIr	10	
				5	14	Irlrr	10	8	36	IrlrIrrl	11	
					15	Irrll	01		37	Irrrllrl	10	
					16	rllrl	11		38	rlrrlrrr	10	
					17	rrIII	10		39	Irrrllrl	11	
									40	rrrllrll	10	

.

both

- (1) Result level 7 will have 6 nodes: IIrIrIr, IrIrIrr, IrrrIIr, rIrrIr, rrIIIr, rrrIII.
- (2) The IIrIrIr result must come from left IIIrr and right Irrl, but the latter doesn't exist.

So this is the end of a false trail, and the path llrlrlr must be removed (i.e., all entries that are prefixes of llrlrlr).

The earlier pos numbers will also change, but we just continue without the llrlrlr entry.

T. H. Merrett

Fixing up the algorithm

By expanding the paths as IrIrIrrl, etc, we have lost all the compression, so just use the original and growing tries instead.

Analyzing the algorithm

Natural join complexity is $\mathcal{O}(n^2)$ for two operands of size n.

So note the double contributions of the operands to the result, in all three phases of the cycle (left, both, right). The doubling starts with a 11 in the common attribute ("both" phase of the cycle), right in cycle 1 for the example. It may double again with further common 11s, and again, and so on. Thus the algorithm is super-linear.

T. H. Merrett