Advanced Data Management

Medha Atre

Office: KD-219
atrem@cse.iitk.ac.in

Aug 11, 2016

Reminder

Assignment-1 due on Aug 15 23:59 IST. Submission instructions will be posted by tomorrow, Friday Aug 12 on the course webpage.

Project groups due on Aug 22, 18:00 IST. Emailing instructions will be given shortly before that.

Recap

- BitMat's query processing algorithm.
- *Semi-joins*.
- Nice properties of *acyclic* query graphs.
- N-way multi-joins.
- Brief introduction to RDF-3X.

Contemporary Systems

- RDF-3X [Neumann2010]
- gStore [Zou2011]
- TripleBit [Yuan2013]
- Virtuoso
- MonetDB
- Neo4j

RDF-3X

- Assumes the graph as a 3-column table.
- Creates all 6-way indexes – PSO, POS, SPO, SOP, OPS, OSP.
- Index compression using *delta-encoding*.
- Indexes are created as compressed B+ trees.
- Creates a *pipelined left-deep* join operator tree.
- *Sideways-information-passing* during scans and merge-joins.
- Aggressive *selectivity estimation* for all possible single edge patterns.

Sideways-information-passing (SIP)

- Passes information of scanned values from one join to another scan, *before* executing that join.
- SIP has a strong similarity to the *semi-joins*.

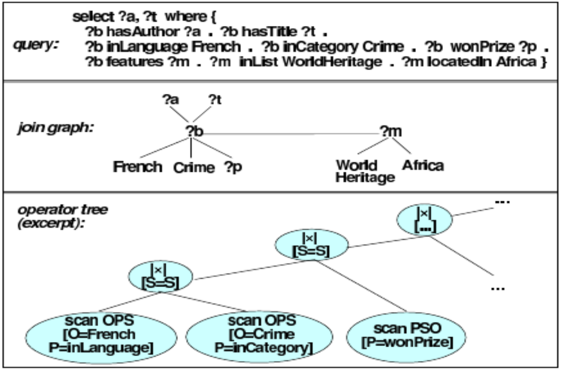


Figure taken from RDF-3X, SIGMOD 2009 paper.

Sideways-information-passing (SIP)

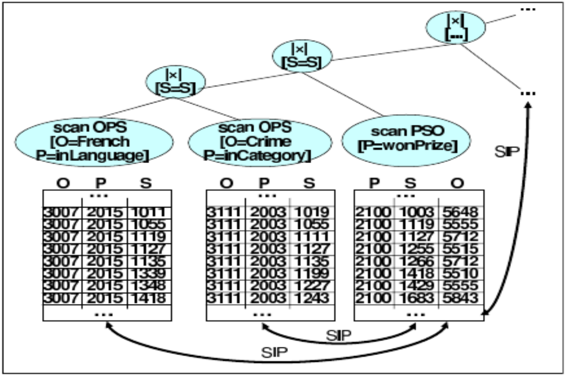


Figure taken from RDF-3X, SIGMOD 2009 paper.

Query processing technique

- Create *equivalence class* of join variables across all the edges in the pattern – inherent to SPARQL, e.g., ?friend, ?sitcom. The equivalence class of ?friend in corresponding SQL query will be {t1.o, t2.s}.
- Create a shared memory for SIP information passing between operators like scan and merge-joins.
- Hash joins created using *bloom filters*.
- Bloom filters are populated with a *distance preserving* hash function.

Query plan

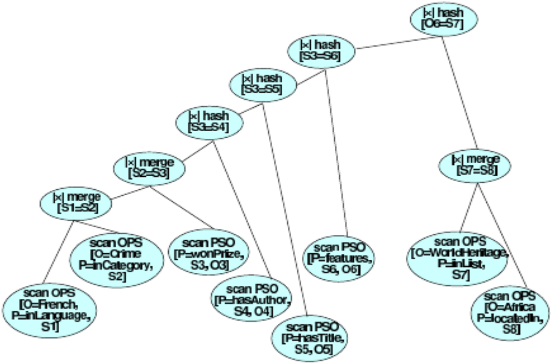


Figure taken from RDF-3X, SIGMOD 2009 paper.

Selectivity estimation

- Creates *aggregation statistics* for *binary projections* and *unary projections*, e.g., for each pair of SP value values of O are indexed along with their frequencies. For each S value, number of edges (tuples) with that value are noted.
- Join selectivities of *pairs* of triple patterns pre-computed.
- The join selectivity estimation for the pairs of triple patterns has a flavor of computing *outgoing and incoming single-hop paths* for all the edges in the graph.

$$sel(c_1, c_2, v) \bowtie_{v=s2} (s_2, p_2, o_2) =$$

$$= \frac{|(c_1, c_2, v) \bowtie_{v=s2} (s_2, p_2, o_2)|}{|(c_1, c_2, v)| * |(s_2, p_2, o_2)|} =$$

$$= \frac{\sum_{x \in \Pi_v(c_1, c_2, v)} |(x, p_2, o_2)|}{|(c_1, c_2, v)| * |(s_2, p_2, o_2)|}$$

Evaluation Results

- Very fast n-way merge-joins through SIP strategy, when there are *star* shaped sub-patterns in the given query.
- Better *selectivity estimation* during query plan generation.
- Fast query evaluation when queries are highly selective and can make use of the optimization techniques of SIP by avoiding large index scans.

TripleBit

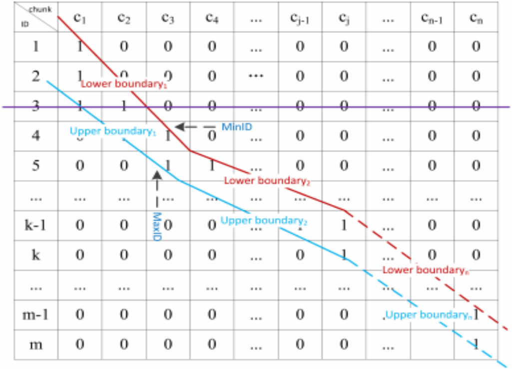
- Uses bitvector based indexing method.
- Compressed *dictionary* encoding.
- Mix of bitvector based indexing (like BitMat) and column-wise storage (like relational column-stores).
- ID-Chunk matrix and ID-Predicate bit matrix.
- ID-predicate index – for edges with wildcards in the query pattern.
- Aggregate indexes – for edges with two wildcards in the query.
- Use technique similar to Byte-aligned Bitmap Compression (BBC).
- Use the *same* query graph model as BitMat!

Triple matrix

	isNamed		hasAuthor		isTitled		hasCitation
	T1	T4	T2	T5	T3	T6	T7
person1	1	0	1	0	0	0	0
person2	0	1	0	1	0	0	0
publication1	0	0	1	0	1	0	1
publication2	0	0	0	1	0	1	1
"Tom"	1	0	0	0	0	0	0
"James"	0	1	0	0	0	0	0
"Pub1"	0	0	0	0	1	0	0
"Pub2"	0	0	0	0	0	1	0

Figure taken from TripleBit, VLDB 2013 paper.

ID-chunk bit matrix



chunk / ID	c_1	c_2	c_3	c_4	...	c_{j-1}	c_j	...	c_{n-1}	c_n
1	1	0	0	0	...	0	0	...	0	0
2	1	0	0	0	...	0	0	...	0	0
3	1	1	0	0	...	0	0	...	0	0
4	0	[illegible]	1	0	...	0	0	...	0	0
5	0	0	1	1	...	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...
k-1	0	0	0	0	...	0	1	...	0	0
k	0	0	0	0	...	0	1	...	0	0
...	...	...	...	...	...	...	...	...	...	...
m-1	0	0	0	0	...	0	0	...	[illegible]	1
m	0	0	0	0	...	0	0	...		1

Figure taken from TripleBit, VLDB 2013 paper.

Byte-aligned Bitmap Compression style

uncompressed (in 7-bit groups)	
A	31 00 03 50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0E
B	56 7F 7F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
compressed	
A	31 81 03 50 8F 0E
B	56 C2 93

Figure taken from TripleBit, VLDB 2013 paper.

Query Processing

- Uses dynamic query plan generation (DQPGA).
- Consists of bushy (star) join processing, similar to *clustered semi-joins* of BitMat, and pruning the data before final joins.

Next Class

General recap of the three systems we learnt, brief overview of other graph related problems, and then moving on to the distributed management.

Questions?