Extended Relational Algebra Operator Reference

Select	$\sigma_c(R)$	$c:$ The selection condition
Extended Project	$\pi_{e_1,e_2,\ldots}(R)$	$e_i:$ The column or expression to project
Product	$R_1 \times R_2$	
Join	$R_1 \bowtie_c R_2$	$c:$ the join condition
Distinct	$\delta(R)$	
Aggregate	$\gamma_{gb_1,gb_2,\ldots,\texttt{AGG}(e_1),\ldots}(R)$	$gb_i:$ group by columns, $e_i:$ expression
Set Difference	$R_1 - R_2$	
Union	$R_1 \cup R_2$	
Sort	τ_A	A one or more attributes to sort on

Relational Algebra Equivalences

Rule	**Notes**
$\sigma_{C_1 \wedge C_2}(R) \equiv \sigma_{C_1}(\sigma_{C_2}(R))$	
$\sigma_{C_1 \vee C_2}(R) \equiv \sigma_{C_1}(R) \cup \sigma_{C_2}(R)$	Note, this is only true for set, not bag union
$\sigma_C(R \times S) \equiv R \bowtie_C S$	
$\sigma_C(R \times S) \equiv \sigma_C(R) \times S$	If C references only R's attributes, also works for joins
$\pi_A(\pi_{A \cup B}(R)) \equiv \pi_A(R)$	
$\sigma_C(\pi_A(R)) \equiv \pi_A(\sigma_C(R))$	If A contains all of the attributes referenced by C
$\pi_{A \cup B}(R \times S) \equiv \pi_A(R) \times \pi_B(S)$	Where A (resp., B) contains attributes in R (resp., S)
$R \times (S \times T) \equiv (R \times S) \times T$	Also works for joins
$R \times S \equiv S \times R$	Also works for joins
$R \cup (S \cup T) \equiv (R \cup S) \cup T$	Also works for intersection and bag-union
$R \cup S \equiv S \cup R$	Also works for intersections and bag-union
$\sigma_C(R \cup S) \equiv \sigma_C(R) \cup \sigma_C(S)$	Also works for intersections and bag-union
$\pi_A(R \cup S) \equiv \pi_A(R) \cup \pi_A(S)$	Also works for intersections and bag-union
$\sigma_C(\gamma_{A,AGG}(R)) \equiv \gamma_{A,AGG}(\sigma_C(R))$	If A contains all of the attributes referenced by C

Cardinality Estimation

Operator	**RA**	**Estimated Size**
Table	R	$\lvert R\rvert$
Projection	$\pi(Q)$	$\lvert Q\rvert$
Union	$Q_1 \uplus Q_2$	$\lvert Q_1\rvert + \lvert Q_2\rvert$
Cross Product	$Q_1 \times Q_2$	$\lvert Q_1\rvert \times \lvert Q_2\rvert$
Sort	$\tau(Q)$	$\lvert Q\rvert$
Limit	$\texttt{LIMIT}_N(Q)$	N
Selection	$\sigma_c(Q)$	$\lvert Q\rvert \times \texttt{SEL}(c, Q)$
Join	$Q_1 \bowtie_c Q_2$	$\lvert Q_1\rvert \times \lvert Q_2\rvert \times \texttt{SEL}(c, Q_1 \times Q_2)$
Distinct	$\delta_A(Q)$	$\texttt{UNIQ}(A, Q)$
Aggregate	$\gamma_{A,B\leftarrow\Sigma}(Q)$	$\texttt{UNIQ}(A, Q)$

Algorithm IO / Memory Costs

Algorithm	IOs Added (pages)	Memory (tuples)
Table Scan	$\frac{\lvert R\rvert}{\mathcal{P}}$	$O(1)$
B+Tree Index Scan	$\log_{\mathcal{I}}(\lvert R\rvert) + \frac{\lvert\sigma_c(R)\rvert}{\mathcal{P}}$	$O(1)$
Hash Index Scan	1	$O(1)$
Projection	0	$O(1)$
Selection	0	$O(1)$
Union	0	$O(1)$
In-Mem Sort	0	$O(\lvert R\rvert)$
2-Pass Sort	$\frac{2\cdot\lfloor log_{\mathcal{B}}(\lvert R\rvert)\rfloor}{\mathcal{P}}$	$O(\mathcal{B})$
In-Mem NLJ	0	$O(\lvert S\rvert)$
On-Disk NLJ	$(1+\lvert R\rvert)\cdot\frac{\lvert S\rvert}{\mathcal{P}}$	$O(1)$
Block-NLJ	$\frac{(1+\lvert R\rvert)}{\mathcal{B}}\cdot\frac{\lvert S\rvert}{\mathcal{P}}$	$O(1)$
1-Pass Hash Join	0	$O(\lvert S\rvert)$
2-Pass Hash Join	$\frac{2\lvert R\rvert+2\lvert S\rvert}{\mathcal{P}}$	$O(1)$
Sort-Merge Join	[As Sort]	$O(Sort)$
B+Tree INLJ	$\lvert R\rvert\cdot(\log_{\mathcal{I}}(\lvert S\rvert)+\frac{\lvert\sigma_c(S)\rvert}{\mathcal{P}})$	$O(1)$
Hash Index NLJ	$\lvert R\rvert\cdot 1$	$O(1)$
Non-Grouped Aggregate	0	$O(1)$
In-Mem Group-By	0	$adom(A)$
Sort Group-By	[As Sort]	$O(Sort)$

Symbol	Meaning
$\mathcal{P}$	Tuples per Page
$\lvert R\rvert$	Size of R
$\mathcal{B}$	Pages of Buffer
$\mathcal{I}$	Keys per Index Page
$adom(A)$	Number of distinct values of attribute A

Part A: ARIES
(15 points)

This question uses the following query log, disk state and checkpoint at the time of a database crash. For each question, state your assumptions and justify your answer.

Log

LSN	Transaction	PrevLSN	Operation
		…	
43	T1	31	$A : 5 \to 10$
44	T2	—	$B : 3 \to 7$
45	T4	—	$A : 10 \to 6$
46	n/a	n/a	`CHECKPOINT`
47	T2	44	$A : 6 \to 7$
48	T3	—	`COMMIT`
49	T1	43	$C : 3 \to 2$
50	T1	49	`COMMIT`
51	n/a	n/a	`CHECKPOINT`
52	T5	—	$C : 2 \to 6$
53	T2	47	$B : 7 \to 4$
54	T4	45	$D : 7 \to 12$

— — **CRASH** — —

Disk State

Object	Value	UpdatedLSN
A	6	45
B	3	29
C	6	52
D	7	36

(Earliest dirty page modified by LSN 44)

Checkpoint @ LSN 46

Transaction	Status	PrevLSN
T1	Running	43
T2	Running	44
T3	Running	—
T4	Running	45

(Earliest dirty page modified by LSN 44)

Checkpoint @ LSN 51

Transaction	Status	PrevLSN
T2	Running	47
T4	Running	45

Question 1. (i) Which log entries are read by the Analyze phase of ARIES. (ii) What is the final state of all in-memory data structures modified during the Analyze phase? **(5 points)**.

Entries 51 to 54 are read during this phase.
The final state of the Transaction Table is:

Transaction	Status	PrevLSN
T2	Running	53
T4	Running	54
T5	Running	52

No other data structures are affected

- (2 points) Entries 51 to 54 are read during this phase. Entries 52, 53, and 54 are also acceptable and got the full credit. Including entries before 51 shows a lack of understanding the Analyze phase, and received no points.

- (3 points) Providing he final state of the Transaction Table. Including other transactions received -1 point penalty. Including the transaction table for log lines 46 to 51 also received -1 point penalty.

Question 2. (i) Which log entries are read by the Redo phase of ARIES. (ii) What is the final state of the 4 database objects at the end of this phase? **(5 points)**.

Full credit: Log entries starting with LSN 44 are read by the Redo phase.
Answers that incorporate either of the following factors are also correct: (1) Log entries 46, 48, 50, 51 have no effect on the Redo Phase, as they only affect the transaction table; (2) Log entries 49 and 52 may be skipped, because the UpdatedLSN of Object C is 52. The final state of the database objects is:
$A = 7$ $B = 4$ $C = 6$ $D = 12$

- (3 points) Entries starting with LSN 44 are read by the REDO phase. Answers that incorporate either of the following factors are also correct: (1) Log entries 46, 48, 50, 51 have no effect on the Redo Phase, as they only affect the transaction table; (2) Log entries 49 and 52 may be skipped, because of UpdatedLSN of Object C is 52. Including log entries that comes before LSN 44 received a -2 point penalty. Not including LSN 44, but giving log entries staring from LSN 45 also received a -2 point penalty.
- (2 points) Providing the final state of the database objects. 2 correct states equal to 1 point. Having one wrong state results in a -1 point penalty.

Question 3. (i) Which log entries are read by the Undo phase of ARIES. (ii) Which log entries are created during this phase. (iii) What is the final state of the 4 database objects at the end of this phase? **(5 points)**.

In order, log entries 54, 53, 52, 47, 45, and 44 are reversed. Each entry creates a corresponding CLR.

LSN	**Transaction**	**PrevLSN**	**Operation**
		...	
55	T4	45	CLR: $D : 7$
56	T2	47	CLR: $B : 7$
57	T5	—	CLR: $C : 2$
58	T2	44	CLR: $A : 6$
59	T4	—	CLR: $A : 10$
60	T2	—	CLR: $B : 3$

The resulting database state is
$A = 10$ $B = 3$ $C = 2$ $D = 7$

- (2 points) In order, log entries 54, 53, 52, 47, 45, and 44 are reversed. Each entry creates a corresponding CLR. Since missing log entries directly affect the output, not including one of these entries result in -2 point penalty. Extra entries, if they affect the output of the Undo phase, result in -2 point penalty. No penalty for giving these log entries in a different order.
- (2 points) Providing the relevant log entries. Including extra log lines such as begin and end transactions are OK as long as the log lines given above appear in the correct order. Missing/Wrong PrevLSN from the logs results in -1 point penalty. Wrong order of the logs such as grouping by transaction results in -2 point penalty.

- (1 point) Providing the final state of the database objects. More than 2 (inclusive) wrong DB objects result in -1 point penalty.

Part B: Transactions
(30 points)

For each of the following schedules indicate by circling which of the following statements is true.

1. The schedule is Conflict Serializable
2. The schedule is View Serializable
3. The schedule could have been created under 2-Phase Locking *without* Deadlock Detection/Avoidance schemes.
4. The schedule could have been created under Normal Timestamp Concurrency Control without triggering an abort.
5. The schedule could have been created under Multi-Version Timestamp Concurrency Control without triggering an abort.

Where relevant, assume that the timestamp of the transaction is the transaction number (e.g., T1 is assigned to timestamp 1).

Question 1. (10 points).

T1	**T2**	**T3**
R(A)		
	W(B)	
		W(B)
R(B)		
W(B)		
	W(C)	
		W(C)
COMMIT		
	COMMIT	
		COMMIT

Conflict	True
View	True
2PL	False
TSCC	False
MVTSCC	True

(circle one in each row)

Question 2. (10 points).

T1	**T2**	**T3**
R(A)		
		W(B)
	W(B)	
R(B)		
W(B)		
	W(C)	
		W(C)
COMMIT		
	COMMIT	
		COMMIT

Conflict	False
View	False
2PL	False
TSCC	False
MVTSCC	True

(circle one in each row)

Question 3. (10 points).

T1	T2	T3
R(A)		
		W(C)
	R(B)	
R(B)		
	COMMIT	
W(B)		
COMMIT		
		COMMIT

Conflict	True
View	True
2PL	True
TSCC	False
MVTSCC	False

(circle one in each row)

Part C: Materialized Views
(20 points)

The following questions are based on the following three tables:

```
CREATE TABLE CUSTOMERS(cust_id int, name varchar, nation_id int);
CREATE TABLE ORDERS(ord_id int, order_date data, cust_id int);
CREATE TABLE LINEITEM(ord_id int, line_num int, part_id int);
```

Each question indicates a table and a query. Assume that the query is materialized in table named `DATA` and that a set of new records to be inserted into the indicated table are available in the table `[tablename]_DELTA` (e.g., `CUSTOMERS_DELTA`).

Write a SQL or Relational Algebra query that *efficiently* computes the new value of `VIEW` after rows are inserted into the indicated table (the delta query).

Question 1. Compute the delta of the following query with respect to rows inserted into `CUSTOMERS`

```
CREATE VIEW DATA AS (
   SELECT * FROM CUSTOMERS c NATURAL JOIN ORDERS o NATURAL JOIN LINEITEM l
   WHERE LINEITEM.part_id = 1 );
```
(5 points).

Unless tables on both sides of a join are being modified simultaneously, the delta of a simple conjunctive query only requires replacing the leaf being modified

```
SELECT * FROM CUSTOMERS_DELTA c NATURAL JOIN ORDERS o NATURAL JOIN LINEITEM l
WHERE LINEITEM.part_id = 1
UNION ALL SELECT * FROM DATA;
```

Grading was as follows:

- -1 if not efficient, should union the delta instead of joining the delta table
- -1 to -3 for missing or wrong tables or query operators

Question 2. Compute the delta of the following query with respect to rows inserted into `LINEITEM`

```
CREATE VIEW DATA AS (
   SELECT * FROM CUSTOMERS c NATURAL JOIN ORDERS o NATURAL JOIN LINEITEM l
   WHERE LINEITEM.part_id = 1 );
```
(5 points).

As before, only one leaf is modified `SELECT * FROM CUSTOMERS c NATURAL JOIN ORDERS o NATURAL JOIN LINEITEM_DELTA`

```
WHERE LINEITEM.part_id = 1
UNION ALL SELECT * FROM DATA;
```

Grading was as follows:

- -1 if not efficient, should union the delta instead of joining the delta table
- -1 to -3 for missing or wrong tables or query operators

Question 3. Compute the delta of the following query with respect to rows inserted into `CUSTOMERS`

```
CREATE VIEW DATA AS (
   SELECT * FROM CUSTOMERS c1 JOIN CUSTOMERS c2 ON nation_id ); (5 points).
```

Distributivity comes in to play with a self-join

```
SELECT * FROM CUSTOMERS_DELTA c1 JOIN CUSTOMERS c2 ON nation_id
UNION ALL
SELECT * FROM CUSTOMERS c1 JOIN CUSTOMERS_DELTA c2 ON nation_id
UNION ALL
SELECT * FROM CUSTOMERS_DELTA c1 JOIN CUSTOMERS_DELTA c2 ON nation_id
```

Grading was as follows:

- -1 to -3 for missing distributed operators
- -5 if they didn't use any distributivity

Question 4. Compute the delta of the following query with respect to rows inserted into `ORDERS`

```
CREATE VIEW DATA AS (
   SELECT COUNT(*) AS cnt FROM CUSTOMERS c NATURAL JOIN ORDERS o NATURAL JOIN LINEITEM l
   WHERE LINEITEM.part_id = 1 ); (5 points).
```

As before... though some trickery is needed to merge the aggregate values. Other answers are also possible.

```
SELECT SUM(cnt) AS cnt FROM (
   SELECT COUNT(*) AS cnt FROM CUSTOMERS c NATURAL JOIN ORDERS_DELTA o NATURAL JOIN LINEITEM l
   WHERE LINEITEM.part_id = 1
   UNION ALL SELECT cnt FROM DATA );
```

Grading was as follows:

- -1 if agg was computed at once instead of using the materialized view and addition (or union)

Part D: Relational Algebra
(15 points)

For each of the following pairs of relational algebra expressions, either prove that they are equivalent by showing a chain of relational algebraic equivalencies, or show a counterexample table where they are not equivalent. State all of your assumptions.

Question 1. (5 points).

$$\pi_{R.A}(\sigma_{R.B=S.B \wedge S.C=T.C \wedge T.D=3}(R \times S \times T)) \overset{?}{\equiv} \pi_{R.A}(R \bowtie_{R.B=S.B} (\pi_{S.B}(S \bowtie_{S.C=T.C} \sigma_{T.D=3}(T))))$$

$$\begin{aligned}
& \pi_{R.A}(\sigma_{R.B=S.B \wedge S.C=T.C \wedge T.D=3}(R \times S \times T)) \\
= \quad & \pi_{R.A}(\sigma_{R.B=S.B \wedge S.C=T.C \wedge T.D=3}(R \times (S \times T))) \\
= \quad & \pi_{R.A}(R \bowtie_{R.B=S.B} (\sigma_{S.C=T.C \wedge T.D=3}(S \times T))) \\
= \quad & \pi_{R.A}(R \bowtie_{R.B=S.B} (S \bowtie_{S.C=T.C} \sigma_{T.D=3}(T))) \\
= \quad & \pi_{R.A}(\pi_{R.A,R.B,S.B}(R \bowtie_{R.B=S.B} (S \bowtie_{S.C=T.C} \sigma_{T.D=3}(T)))) \\
= \quad & \pi_{R.A}(\pi_{R.A,R.B}(R) \bowtie_{R.B=S.B} (\pi_{S.B}(S \bowtie_{S.C=T.C} \sigma_{T.D=3}(T)))) \\
= \quad & \pi_{R.A}(\pi_{R.A,R.B}(R) \bowtie_{R.B=S.B} (\pi_{S.B}(\pi_{S.B}(S \bowtie_{S.C=T.C} \sigma_{T.D=3}(T))))) \\
= \quad & \pi_{R.A}(\pi_{R.A,R.B,S.B}(R \bowtie_{R.B=S.B} (\pi_{S.B}(S \bowtie_{S.C=T.C} \sigma_{T.D=3}(T))))) \\
= \quad & \pi_{R.A}(R \bowtie_{R.B=S.B} (\pi_{S.B}(S \bowtie_{S.C=T.C} \sigma_{T.D=3}(T))))
\end{aligned}$$

Not every step needed to be given, points were awarded as follows:

- 2 points for converting cross-products to joins (Equations **??** and **??**)
- 1 point for splitting up conjuctions in σ (Equation **??**)
- 1 point for filter push down (Equation **??**)
- 1 point for mentioning projection duplication (Equation **??**). Just mentioning the rule was sufficient for full credit.

Question 2. (5 points).

$$(R \cup R')\cancel{\bowtie_{R.A=S.A}} \times (S \cup S') \overset{?}{\equiv} (R \times S) \cup (R \times S') \cup (R' \times S) \cup (R' \times S')$$

$$\begin{aligned}
& (R \cup R') \times (S \cup S') \\
= \quad & (R \times (S \cup S')) \cup (R' \times (S \cup S')) \\
= \quad & (R \times S) \cup (R \times S') \cup (R \times S) \cup (R \times S')
\end{aligned}$$

- 5 points for any answer that mentioned or showed the distributive law.

- -2 points for not changing the $\bowtie_{R.A=S.A}$ to a $\times$ (errata discussed and posted during the exam)

Question 3. (5 points).

$$\sigma_{TOT>20}(\gamma_{R.A,TOT\leftarrow SUM(R.B)}(R)) \overset{?}{\equiv} \gamma_{R.A,TOT\leftarrow SUM(R.B)}(\sigma_{R.B>20}(R))$$

Not equivalent. Consider the following table:

R	**A**	**B**
	1	15
	1	16

The left-hand-side expression produces a single tuple: $\langle A:1, TOT:31\rangle$, while the right hand side produces no results.

- 5 points for a counterexample

Part E: Potpouri
(20 points)

Answer each of the following questions true or false.

In a typical database buffer manager, a page is pinned while a transaction is using it. True

A typical database buffer manager is a specialized form of an operating system's virtual memory manager. True

A bloom filter requires a *linear* amount of space in the number of entries in the filter. True

A lower bound on the amount of data that must be transmitted during a semi-join is the number of distinct values of the join attribute in either of the two tables being joined. True

For any join of the form $R \bowtie_{R.A\ \theta\ S.B} S$ where θ is one of $<$, $>$, $\geq$, $\leq$, or $=$, and where there are N range-based partitions of R on $R.A$ and M range-based partitions of S on $S.B$, every single pair of partitions is capable of producing join results. False

Roughly half of the partition pairs will not produce any results for $\theta \neq$ '$=$', and roughly $O(M + N)$ will produce results for an equijoin.

A bloom filter can tell you if an element is definitely *not* in a set, but it can not tell you that an element is definitely in a set. True

The number of pages required by a *dynamic* hash table is always at least half of the number of entries in its directory table. False

A dynamic hash table does not require this. A dynamic hash table behaves like at tree, splitting hash buckets whenever one fills up. In the extreme case, all values match on the d least significant bits, requiring requiring that that this one bucket be split d times. This results in a directory of size 2^d (since the directory requires one entry for each potential bucket that could exist for that many splits), but the number of pages consumed will only be $2d$ or $O(\log_2(2^d))$

Under two phase commit, each transaction can have its own coordinator. True

Under two phase commit, once a participating node sends a `COMMIT` message to the coordinator, it can not allow a different, conflicting transaction to commit. True

A cycle in the waits for graph indicates a deadlock. True