CS 186 Discussion 9:
Query Optimization and Database Design

1 Selectivity Estimation

Consider a relation $R(a, b, c)$ with 1000 tuples. We have an index on a with 50 unique values in the range $[1, 50]$ and an index on b with 100 unique values in the range $[1, 100]$. We do not have an index on c. Use selectivity estimation to estimate the number of tuples produced by the following queries.

1. `SELECT * FROM R` 1000
2. `SELECT * FROM R WHERE a = 42` $\frac{1}{50} \cdot 1000 = 20$
3. `SELECT * FROM R WHERE b = 42` $\frac{1}{100} \cdot 1000 = 10$
4. `SELECT * FROM R WHERE c = 42` $\frac{1}{10} \cdot 1000 = 100$
5. `SELECT * FROM R WHERE a <= 25` $\frac{1}{2} \cdot 1000 = 500$
6. `SELECT * FROM R WHERE b <= 25` $\frac{1}{4} \cdot 1000 = 250$
7. `SELECT * FROM R WHERE c <= 25` $\frac{1}{10} \cdot 1000 = 100$
8. `SELECT * FROM R WHERE a <= 25 AND b <= 25` $\frac{1}{2} \cdot \frac{1}{4} \cdot 1000 = 125$
9. `SELECT * FROM R WHERE a <= 25 AND c <= 25` $\frac{1}{2} \cdot \frac{1}{10} \cdot 1000 = 50$
10. `SELECT * FROM R WHERE a <= 25 OR b <= 25` $(\frac{1}{2} + \frac{1}{4} - \frac{1}{2} \cdot \frac{1}{4}) \cdot 1000 = 625$
11. `SELECT * FROM R WHERE a = b` $\frac{1}{100} \cdot 1000 = 10$
12. `SELECT * FROM R WHERE a = c` $\frac{1}{50} \cdot 1000 = 20$

2 Query Optimization

Consider the relations $R(a, b)$, $S(b, c)$, and $T(c, d)$ with clustered indexes on $R.a$, $S.b$, and $T.c$ and with unclustered indexes on $S.c$ and $T.d$. All indexes have index keys in the range $[1, 100]$. We want to optimize the following query:

```
SELECT *
 FROM  R, S, T
WHERE  R.b = S.b AND S.c = T.c
  AND  R.a <= 50
  AND  (T.c <= 50 AND T.d <= 20)
```

In the first pass of the Sellinger query optimization algorithm, we compute the minimum cost access method for every (relation, interesting order) pair. Complete the following table which computes this. Let $[R]$, $[S]$, and $[T]$ be the number of *pages* in R, S, and T. Similarly, let $|R|$, $|S|$, and $|T|$ be the number of *tuples* in R, S, and T.

Relation	Access Method	Interesting Order	I/O Cost	Output Size	Retained
R	Full table scan	None	$[R]$	$0.5[R]$	No
	Index scan on a	(a)	$2 + 0.5[R]$		Yes
S	Full table scan	None	$[S]$	$[S]$	Yes
	Index scan on b	(b)	$2 + [S]$		Yes
	Index scan on c	(c)	$2 + \|S\|$		Yes
T	Full table scan	None	$[T]$	$0.1[T]$	No
	Index scan on c	(c)	$2 + 0.5[T]$		Yes
	Index scan on d	(d)	$2 + 0.2\|T\|$		Yes

In the second pass of the Sellinger query optimization algorithm, we consider each (relation, interesting order) pair in turn. For each, we compute the cost of joining every other relation into it. In the end, we retain the minimum cost join for each (relations, interesting order) pair. Complete the following table which computes *part* of the second pass. Let B be the number of buffer pages, let $SC(x) = 2x\lceil 1 + \log_{B-1}(\lceil \frac{x}{B} \rceil)\rceil$, assume that a B+ tree index can be traversed in 2 IOs, and assume the merge phase of a sort merge join on two relations of size n and m can be done in $n + m$ IOs. Note that we do not consider joining R and T because we favor joins over cross products.

Left Relations	Left Interesting Order	Right Relation	Join	Interesting Order	I/O Cost	Output Size
R	(a)	S	BNLJ	None	$0.5[R] + \left\lceil \frac{0.5[R]}{B-2} \right\rceil [S]$	$\frac{0.5[R][S]}{100}$
			SMJ	(b, a)	$\text{SC}(0.5[R]) + 0.5[R] + [S]$	
S	None	R	BNLJ	None	$[S] + \left\lceil \frac{[S]}{B-2} \right\rceil [R]$	$\frac{0.5[R][S]}{100}$
			SMJ	(b)	$\text{SC}([S]) + \text{SC}(0.5[R]) + [S] + 0.5[R]$	
S	None	T	BNLJ	None	$[S] + \left\lceil \frac{[S]}{B-2} \right\rceil [T]$	$\frac{0.1[S][T]}{100}$
			SMJ	(c)	$\text{SC}([S]) + [S] + 0.1[T]$	
S	(b)	R	BNLJ	None	$[S] + \left\lceil \frac{[S]}{B-2} \right\rceil [R]$	$\frac{0.5[R][S]}{100}$
			SMJ	(b)	$SC(0.5[R]) + [S] + 0.5[R]$	
S	(b)	T	BNLJ	None	$[S] + \left\lceil \frac{[S]}{B-2} \right\rceil [T]$	$\frac{0.1[S][T]}{100}$
			SMJ	(c, b)	$\text{SC}([S]) + [S] + 0.1[T]$	

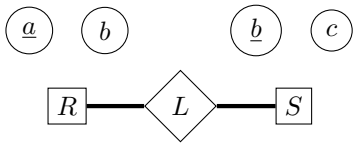
3 ER Diagrams

Consider the following ER diagrams. We can translate each into an equivalent triplet of `CREATE TABLE` statements: one for R, one for S, and one for T. The statements for R and S remain constant:

```
CREATE TABLE R (a INTEGER, b INTEGER, PRIMARY KEY (a))
CREATE TABLE S (b INTEGER, c INTEGER, PRIMARY KEY (b))
```

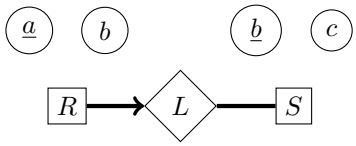
Provide the `CREATE TABLE` statements for L.

1.



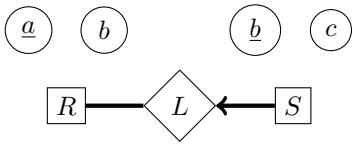

```
CREATE TABLE L(
  a INTEGER, b INTEGER,
  FOREIGN KEY (a) REFERENCES R,
  FOREIGN KEY (b) REFERENCES S
)
```

2.




```
CREATE TABLE L(
  a INTEGER, b INTEGER,
  PRIMARY KEY (a)
  FOREIGN KEY (a) REFERENCES R,
  FOREIGN KEY (b) REFERENCES S
)
```

3.




```
CREATE TABLE L(
  a INTEGER, b INTEGER,
  PRIMARY KEY (b),
  FOREIGN KEY (a) REFERENCES R,
  FOREIGN KEY (b) REFERENCES S
)
```