16. Review 2 Exercises
Problem: Table: R(A,B,C,D,E)
F= \{ \{A\} \rightarrow \{B, C\}, \{B\} \rightarrow \{A, C\}, \{A, D\} \rightarrow \{E\}, \{E\} \rightarrow \{D\}\}

1] Which of the following sets of functional dependencies is a canonical cover of F?

a) \{\{A\} \rightarrow \{B, C\}, \{B\} \rightarrow \{A,C\}, \{A,D\} \rightarrow \{E\}, \{E\rightarrow D\}\}

b) \{\{A\} \rightarrow \{B\}, \{A\} \rightarrow \{C\}, \{B\} \rightarrow \{C\}, \{A,D\} \rightarrow \{E\}\}

c) \{\{A\} \rightarrow \{B,C\}, \{B\} \rightarrow \{A\}, \{A,D\} \rightarrow \{E\}, \{E\} \rightarrow \{D\}\}

d) \{\{A\} \rightarrow \{B\}, \{B\} \rightarrow \{A,C\}, \{A,D\} \rightarrow \{E\}, \{E\} \rightarrow \{D\}\}

e) Both c) and d)

2] Which of the following functional dependencies is not in the closure F+?

a) \{A\} \rightarrow \{B\}

b) \{B\} \rightarrow \{B, C\}

c) \{C\} \rightarrow \{A\}

d) \{A, D\} \rightarrow \{C, E\}

e) \{A\} \rightarrow \{C\}
Table: R(A,B,C,D,E)
F= \{ \{A\} \rightarrow \{B, C\}, \{B\} \rightarrow \{A, C\}, \{A, D\} \rightarrow \{E\}, \\
\{E\} \rightarrow \{D\} \}\n
3] Which of the following set is a subset of \{A, D\}+?
   a) \{A, B\}
   b) \{B, C, D\}
   c) \{E\}
   d) All of the above
   e) None of the above

4] Which of the following is a superkey for R?
   a) \{A\}
   b) \{AB\}
   c) \{BC\}
   d) \{ACD\}
   e) \{ABC\}
Table: R(A,B,C,D,E)
F = \{ \{A\} \rightarrow \{B, C\}, \{B\} \rightarrow \{A, C\}, \{A, D\} \rightarrow \{E\}, \{E\} \rightarrow \{D\} \}\n
5] Which of the following is a candidate key for R?
   a) AD
   b) AE
   c) BD
   d) BE
   e) All of the above

6] Consider the following decomposition: \{A, B, C\}, \{A, D, E\}. Which of the following statements is true?
   a) The decomposition is 3NF, lossless join and dependency preserving
   b) The decomposition is 3NF, lossless join but not dependency preserving
   c) The decomposition is 3NF, dependency preserving, but not lossless join
   d) The decomposition is lossless join, dependency preserving but not 3NF
   e) The decomposition is 3NF, but neither lossless join nor dependency preserving
Consider the following decomposition: \{A, B, C\}, \{A, E\}, \{D, E\}. Which of the following statements is true?

a) The decomposition is BCNF, lossless join and dependency preserving

b) The decomposition is BCNF, lossless join but not dependency preserving

c) The decomposition is BCNF, dependency preserving, but not lossless join

d) The decomposition is lossless join, dependency preserving but not BCNF

e) The decomposition is BCNF, but neither lossless join nor dependency preserving
Problem – Normal Forms

- Consider table R(A, B, C, D, E). Given the functional dependencies in the first column of the following form, complete the form accordingly.
- In the second column, list all candidate keys for R.
- In the third column, provide a maximal decomposition of R in 3NF (we only decompose when there is violation of 3NF) – if R is already in 3NF just write R(A, B, C, D, E) instead of a decomposition.
- In the fourth column, do the same for BCNF decomposition. If there are multiple options, choose any dependency preserving decomposition.
- Give the results without comments. The first row is given as an example. Each answer is 2%.

<table>
<thead>
<tr>
<th>Functional Dependencies</th>
<th>Candidate Keys for R</th>
<th>Decompose R in 3NF</th>
<th>Decompose R in BCNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A \rightarrow B, C, D, E}</td>
<td>{A}</td>
<td>R(A, B, C, D, E)</td>
<td>R(A, B, C, D, E)</td>
</tr>
</tbody>
</table>
## Problem – Normal Forms

<table>
<thead>
<tr>
<th>Functional Dependencies</th>
<th>Candidate Keys for R</th>
<th>Decompose R in 3NF</th>
<th>Decompose R in BCNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ C → D }</td>
<td>ABCE</td>
<td>R1(A,B,C,E)</td>
<td>R1(A,B,C,E)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R2(C,D)</td>
<td>R2(C,D)</td>
</tr>
<tr>
<td>{ C→D, D→C }</td>
<td>ABCE, ABDE</td>
<td>R(A,B,C,D,E)</td>
<td>R1(C,D)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R2(A,B,C,E) or R1(C,D)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R2(A,B,D,E)</td>
</tr>
<tr>
<td>{ A → B, C→D}</td>
<td>ACE</td>
<td>R1(A,B)</td>
<td>R1(A,B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R2(C,D)</td>
<td>R2(C,D)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R3(A,C,E)</td>
<td>R3(A,C,E)</td>
</tr>
<tr>
<td>{ A → BC, D → AE}</td>
<td>D</td>
<td>R1(A,B,C)</td>
<td>R1(D,A,E)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R2(D,A,E)</td>
<td>R2(A,B,C)</td>
</tr>
</tbody>
</table>
Consider the file

- Sailors (sname, sid, age, rating)

\( n_{\text{sailors}} = 10,000 \) records, \( b_{\text{sailors}} = 1,000 \) pages (10 records/page)

\( V(\text{rating}, \text{Sailors}) = 10 \) and \( V(\text{age}, \text{Sailors}) = 100 \)

- Query:

  ```sql
  SELECT name
  FROM Sailors
  WHERE rating = 7 AND age = 40
  ```

Describe alternative plans to process the query and estimate their cost assuming uniformity and attribute independence.

How many records you expect in the result:

\[ 10,000 \times \frac{1}{10} \times \frac{1}{100} = 10 \text{ records} \]
Single Relation Plans

1] **Sequential (Linear) Scan.** Read the whole Sailors file and select the records that satisfy both conditions.
   
   Cost = \( b_{\text{Sailors}} = 1,000 \) pages

2] **Binary Search.** If the file is sorted on Rating (or Age), do binary search and find the first record satisfying the condition Rating=7 (or Age=40). Retrieve the remaining records sequentially and filter out non qualifying tuples.

   • *Question* What is cheaper: Sorting on Rating or Age?

   Cost for sorting on Rating = \( \log_2 1000 + 99 = 109 \)

   Cost for sorting on Age = \( \log_2 1000 + 9 = 19 \)

3] **Single-index.** Use an index for one of the two conditions and check the other condition in memory: *if we have an index on Rating we use this index to find records where Rating=7. For each such record we check the condition Age=40.*

   • Not necessarily good solution if index is not clustered. There are 1000 sailors with Rating=7. If index on Rating is not clustered, we need 1,000 random page accesses to retrieve these sailors. Better to do sequential scan.

   • If the index is clustered, all sailors with Rating=7 are in 100 consecutive pages.
Additional Single Relation Plans

4] **Multidimensional index.** If we have a multidimensional index (e.g., Grid file) on rating and age we can use this index to retrieve the records that satisfy both conditions. **Cost depends on the index.**

5] **Multiple-indexes.** Use more than one index and do intersection of record pointers (Rids) before retrieval of actual records: *use the index on Age and find all Rids of records where Age=40. Then use an index on Rating to find all Rids where Rating=7. Find the intersection of these two sets of Rids (using sorting) and only retrieve records that belong to the intersection. Applicable only if both indexes contain record pointers.*

6] **Covering Index.** If all attributes mentioned in the query match a dense index we can do an index only scan without accessing the actual file. *If, for instance, we have a B+-tree on <name, age, rating> we can answer the query by doing a sequential scan of the index. If the B+-tree is on <age, rating, name> we only need to read the part of the index for age = 40 and rating = 7.*
Multi-Relation Plans - Estimation of Output Size

Assume that there are 10,000 Sailors, 100,000 Reservations, 1,000 Boats and $V(\text{date, Reserves})=1000$, $V(\text{color, Boat})=10$

What is the expected number of records in the result of the query?

```sql
SELECT *
FROM Sailor S, Reserves R, Boats B
```

There are 100 reservations ($n_{\text{reserves}}/V(\text{date, Reserves})$) on 1.1.2005. 10% of these reservations (i.e., 10) are on red boats. Thus, the expected result contains 10 records.

Alternative solution by estimating the join result before applying selections:

- S JOIN R contains 100,000 records
- (S JOIN R) JOIN B contains 100,000 records
- The output size of $\sigma_{\text{R.date}=1.1.2005 \text{ and B.color}=\text{red}}$ (S JOIN R JOIN B) is $100,000 \times \text{Selectivity}_{\text{R.date}=1.1.2005} \times \text{Selectivity}_{\text{color}=\text{red}} = 100,000 \times 1/1000 \times 1/10 = 10$
Evaluation/execution Plans

Assume that for all files there are 10 records per page and you have the following indexes.

1] Hash index on S.sid for Sailors (no overflow buckets)
2] Clustered B+-tree on R.date for Reserves (2 levels)
3] Hash index on B.bid for Boats (no overflow buckets)

Describe different plans for processing the query and estimate their cost.

Alternative join orders after heuristic optimization (pushing the selections down)

1] (Sailor JOIN \( \sigma_{\text{R.date}=1.1.2005} \) Reserves) JOIN \( \sigma_{\text{color}=\text{red}} \) Boats
2] Sailor JOIN (\( \sigma_{\text{R.date}=1.1.2005} \) Reserves JOIN \( \sigma_{\text{color}=\text{red}} \) Boats)
Total cost: $C_1 + C_2 + C_3$

$C_1$: Cost of computing $\text{Temp1} = (\text{Sailor JOIN } \sigma_{R.\text{date}=1.1.2005} \text{Reserves})$

$C_2$: Cost of computing $\text{Temp2} = \sigma_{\text{color}=\text{red}} \text{Boats}$

$C_3$: Cost of $\text{Temp1 JOIN Temp2}$

In order to estimate $C_1$, we need to determine the best sub-plan $P_1$ for computing $\text{Temp1}$

Some alternatives:
1. BNL using Sailor as the outer relation
2. BNL using Reserves as the outer relation
3. Sort-merge join (we have to sort both tables on sid)
4. Hash join
5. Index nested loop with Reserves as the outer relation. Sailors contains a hash index on the join attribute sid. Furthermore, we have a selective condition $(\sigma_{R.\text{date}=1.1.2005} \text{Reserves})$ and a clustering index on $\text{Reserves.date}$.

Best Option
Estimation of $C_1$
$\text{Temp1} = \text{Sailor JOIN } \sigma_{R.\text{date}=1.1.2005} \text{Reserves}$

**Sub-Plan P1**: Use clustering B+-tree on Reserves.date to find all reservations on 1.1.2005.

How many?

$n_{\text{Reserves}}/V(\text{Date,Reserves})=100,000/1,000=100$

How many pages do I need to access, in order to find these reservations: $2+10$ (2 in the index and 10 in the file – the reservations are ordered on the date)

For each reservation, I retrieve the corresponding sailor record using the hash index on Sailor.sid, with cost 2 per reservation.

Total cost:

$12+100*2=212$

- Do I need to consider other algorithms that produce results in interesting orders?

In this case **no**, because the only useful order would be on the Reserves.bid, which cannot be generated by any algorithm in this example.
Estimation of $C_2 - \text{Temp2} = \sigma_{\text{color}=\text{red}} \text{Boats}$ and $C_3 - \text{Temp1 JOIN Temp2}$

$C_2$ using Sub-Plan $P_2$: For Boats, I only have a hash index on bid (no index on color). Therefore, in order to find red boats, I need to scan the entire (1,000 boat records) file with cost $b_{\text{Boats}} = 100$ pages. Since only 10% of the boats are red, I expect to retrieve 100 records.

$C_3$ using Sub-Plan $P_3$ (materialization). Store the intermediate results of $P_2$ and $P_1$, then read them and join them on the bid using any join algorithm. Very expensive.

$C_3$ using Sub-Plan $P_3$ (pipelining). Recall that during $P_1$, we generate 100 (Sailor JOIN $\sigma_{\text{R.date}=1.1.2005} \text{Reserves}$) records. When each such record is generated, we find the corresponding Boat, using the hash index on Boats.bid with cost 2. If the color is not red I discard the record (i.e., I perform the selection on the fly without the need for $P_2$). This corresponds to the Index Nested Loops algorithm and has cost $100 \times 2$.

Total cost for 1st expression: $C_1 + C_3 = 212 + 200 = 412$. 

15
Evaluation plan for 1^{st} expression
\((\text{Sailor JOIN } \sigma_{\text{R.date}=1.1.2005} \text{Reserves}) \text{ JOIN } \sigma_{\text{color}=\text{red}} \text{Boats})\)

1. Use B+-tree on R.date
   - Will retrieve 100 records with cost 2+10

2. Index nested loop for every reservation on 1.1.2005
   - Use hash index on S.sid to find information about the sailor
   - Cost 100(1+1)
   - Output contains 100 records

3. Index nested loop for every record of step 2
   - Use hash index on B.bid to find information about the boat
   - Filter B.color condition in memory
   - The cost is 1+1 per reservation, i.e., total cost 200 for all reservations of step 1
   - The output contains only 10 records (10% of the reservations are on red boats)

Total cost = 12+200+200=412
Evaluation plan for 2nd expression
Sailor JOIN (σ_{R.date=1.1.2005} Reserves JOIN σ_{color=red} Boats) using the same concepts

Total cost = 12 + 200 + 20 = 232

3. Index nested loop
   for every reservation that satisfies steps 1 and 2
   use hash index on S.sid to find information about the sailor
   The cost is 1+1 per reservation, i.e., total cost 20 for all reservations of step 2

2. Index nested loop
   for every reservation that satisfies the date condition
   use hash index on B.bid to find information about the boat
   filter B.color condition in memory
   The cost is 1+1 per reservation, i.e., total cost 200 for all reservations of step
   The output contains only 10 records (10% of the reservations are on red boat):

1. Use B+-tree on R.date
   Will retrieve 100 records
   with cost 2+10

Reserves R

Boats B
The optimizer will choose to execute the last plan, i.e., for Sailor JOIN (\(\sigma_{R.\text{date}=1.1.2005}\) Reserves JOIN \(\sigma_{\text{color}=\text{red}}\) Boats), because it is cheaper. Intuitively, this is expected because this expression first performs the joins on the tables that involve selections. Real optimizers follow similar principles but include additional factors such as CPU time, and the difference between random and sequential I/O operation.
Is the schedule R2(A) R1(A) W1(A) W2(B) C2 C1 recoverable and cascadeless?
Both – no transaction reads items written by the other.

Rewrite the schedule R2(A) R1(A) W1(A) W2(B) according to 2PL protocol (i.e., add **lock-S, lock-X, unlock** statements below). Explain briefly whether the schedule will terminate or fail and why?

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lock_S(A)</td>
</tr>
<tr>
<td>lock_S(A)</td>
<td>READ(A)</td>
</tr>
<tr>
<td>READ(A)</td>
<td></td>
</tr>
<tr>
<td>lock_X(A) - cannot</td>
<td>WRITE(A)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WRITE(B)</td>
</tr>
</tbody>
</table>
• Rewrite the schedule R2(A) R1(A) W1(A) W2(B) according to **TS-ordering** protocol (i.e., add **read_TS** and **write_TS** statements) assuming that the timestamps of T1, T2 are 2, 1, respectively. The initial timestamps of A and B are 0. Explain briefly whether the schedule will terminate or fail and why?

<table>
<thead>
<tr>
<th>T1, timestamp=2</th>
<th>T2, timestamp=1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>READ(A)</td>
</tr>
<tr>
<td></td>
<td>read_TS(A)=1</td>
</tr>
<tr>
<td>READ(A)</td>
<td></td>
</tr>
<tr>
<td>read_TS(A)=2</td>
<td>WRITE(A)</td>
</tr>
<tr>
<td>write_TS(A)=2</td>
<td>WRITE(B)</td>
</tr>
<tr>
<td></td>
<td>write_TS(B)=1</td>
</tr>
</tbody>
</table>
Multiversion Timestamp Protocol

- Rewrite the schedule R2(A) R1(A) W1(A) W2(B) according to multiversion **TS-ordering** protocol (i.e., add **read_TS** and **write_TS** statements and specify the versions of the items) assuming that the timestamps of T1, T2 are 1, 2, respectively and that the initial versions of items are A0, B0. Complete the correct version numbers (e.g., READ(A0) instead of READ(A)).

<table>
<thead>
<tr>
<th>T1, timestamp=1</th>
<th>T2, timestamp=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ(A0)</td>
<td>READ(A0)</td>
</tr>
<tr>
<td>read_TS(A0)=2</td>
<td>read_TS(A0)=2</td>
</tr>
<tr>
<td>READ(A0)</td>
<td>WRITE(A1)</td>
</tr>
<tr>
<td>read_TS(A0)=2</td>
<td>failure</td>
</tr>
<tr>
<td>WRITE(B)</td>
<td>WRITE(B)</td>
</tr>
</tbody>
</table>