

DBS Exam Notes

January 12, 2026

Contents

1 Functional Dependencies	2
1.1 The "Boss and Follower" Logic	2
1.2 Verification Results	2
2 Finding Superkeys	2
2.1 Step-by-Step Process	2
3 Normal Form Audit	3
3.1 BCNF Check	3
3.2 3NF Check	3
4 Lossless Join	3
5 External Merge Sort: 2-Way Algorithm	3
5.1 Problem Example	3
5.2 Step-by-Step Breakdown	3
5.3 Why the Other Options Are Wrong	4
6 Multi-Way External Merge Sort	4
6.1 Example: Using Relation r_1	4
6.2 Quick Comparison	5
7 Page-Oriented Nested-Loop Joins	5
7.1 Task-by-Task Logic	5

1 Functional Dependencies

1.1 The "Boss and Follower" Logic

A functional dependency $\alpha \rightarrow \beta$ is a rule stating that if two rows have the same value for the "Boss" (α), they **must** have the same value for the "Follower" (β).

Case Study: Slide 16 Instance

Based on the table instance provided on Slide 16 of Lecture 7, here is the verification for each functional dependency using the step-by-step process:

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d2
a2	b2	c2	d3
a3	b1	c3	d3

1.2 Verification Results

- $A \rightarrow B$ (**Holds**):
 1. **Step-by-Step Check:** Identify the Boss (A) and Follower (B).
 2. **Find Duplicates:** Look for rows where the Boss (A) has the same value.
 3. **Check Followers:** For each duplicate Boss, verify that the Follower (B) values are identical.
 4. **Verification:** The only duplicate "Boss" in column A is **a2** (rows 2 and 3).
 5. **Verdict:** In both rows, the "Follower" in column B is **b2**. Since the followers are identical for the duplicate boss, the dependency holds.
- $A \rightarrow C$ (**Holds**):
 - For the duplicate boss **a2**, both rows have the identical follower **c2** in column C .
- $A \rightarrow D$ (**Fails**):
 - For the duplicate boss **a2**, the followers in column D are **d2** and **d3**.
 - Because the followers are different for the same boss, the dependency is broken.

2 Finding Superkeys

2.1 Step-by-Step Process

1. Start with a candidate attribute (e.g., B)
2. Check if B is a "Boss" for any functional dependency rules
3. If $B \rightarrow A$, your set becomes $\{B, A\}$
4. Continue using your new set to unlock more attributes
5. If $\{A, B\}$ is now in your set and $AB \rightarrow C$, you add C
6. If you reach all attributes $\{A, B, C, D, E, F\}$, it is a superkey

3 Normal Form Audit

3.1 BCNF Check

- Look at every functional dependency
- Is the "Boss" (left side) a superkey?
- If even one is not, it is **NOT in BCNF**

3.2 3NF Check

- If BCNF fails, check the "Follower" (right side)
- Is it a prime attribute (part of any candidate key)?
- If yes, it is **3NF**

4 Lossless Join

A split into R_1 and R_2 is **lossless** if the attributes they share are a superkey for at least one of the two resulting tables.

5 External Merge Sort: 2-Way Algorithm

5.1 Problem Example

The answer to Question 6.1 is $\lceil \log_2 2,000 \rceil$ because the algorithm must first convert the raw data into manageable pages and then iteratively merge those pages until they are sorted.

5.2 Step-by-Step Breakdown

Step 1: Calculate the Number of Pages (B)

The algorithm operates on pages (blocks), not individual records.

- Total Tuples (n_{r1}): 100,000
- Tuples per Page: 50
- Total Pages (B): $\frac{100,000}{50} = 2,000$ pages

Step 2: Understand the Sorting Phases

External sorting is divided into two distinct phases:

- **Phase 1 (Pass 0):** The database reads each page into memory, sorts it, and writes it back to disk. This creates 2,000 sorted runs, each consisting of 1 page.
- **Phase 2 (The Merge Phase):** This is what the question specifically asks for. In this phase, the algorithm takes the sorted runs and merges them into larger and larger runs.

Step 3: Apply the 2-Way Merge Logic

In a 2-Way merge, the computer uses 3 buffer pages: two for input (to read two runs) and one for output (to write the merged result).

The Power of 2: Because it is a "2-Way" merge, it combines 2 runs into 1 larger run during every pass.

- Pass 1: 2,000 runs are merged into 1,000 runs
- Pass 2: 1,000 runs are merged into 500 runs
- **Goal:** This continues until only 1 single sorted run remains

Step 4: The Mathematical Formula

To find out how many times you must halve the number of runs to reach 1, you use a logarithm with base 2.

Formula for Phase 2 Passes: $\lceil \log_2(\text{Initial Runs}) \rceil$

Since Phase 1 produced 2,000 runs, Phase 2 requires $\lceil \log_2 2,000 \rceil$ passes.

5.3 Why the Other Options Are Wrong

- (a) $\lceil \log_2 100,000 \rceil$: This uses the number of tuples, but the database sorts pages.
- (c) & (d): These use a base of 299 ($M - 1$), which is the formula for a Multi-Way Merge Sort using all 300 buffer pages, but the question explicitly asked for the 2-Way algorithm.

6 Multi-Way External Merge Sort

For Multi-Way External Merge Sort, the logic shifts from merging only two runs at a time to using almost all available buffer pages to merge as many runs as possible in a single pass.

6.1 Example: Using Relation r_1

Using the same relation r_1 from Question 6.1 (with 2,000 pages and 300 buffer pages) as an example:

1. Calculate Initial Runs (Phase 1 / Pass 0)

- Total Pages (B): 2,000
- Buffer Pages (M): 300
- In Phase 1, we read M pages at a time, sort them in memory, and write them out as a "run"
- **Number of Initial Runs:** $\lceil B/M \rceil = \lceil 2,000/300 \rceil = 7$ runs

2. The Multi-Way Merge Logic (Phase 2)

In a Multi-Way merge, you use $M - 1$ buffer pages as input buffers (to read from $M - 1$ different runs simultaneously) and 1 page as an output buffer.

- **Fan-in ($M - 1$):** $300 - 1 = 299$. This means you can merge up to 299 runs into 1 larger run in a single pass.

3. The Mathematical Formula

To find the number of passes required in Phase 2 to reach a single sorted file, use the formula:

$$\lceil \log_{M-1}(B/M) \rceil$$

4. Applying it to the Example

- **Phase 1 Cost:** $2 \times B = 4,000$ I/Os (reading and writing all pages once)
- **Phase 2 Passes:** $\lceil \log_{299}(7) \rceil$. Since 7 is much smaller than 299, it only takes 1 pass to merge all runs into the final sorted file.
- **Total I/O Cost:** $2 \times B \times (1 + \text{number of passes in Phase 2})$. For this specific case: $2 \times 2,000 \times (1 + 1) = 8,000$ I/Os.

6.2 Quick Comparison

Feature	2-Way Merge Sort	Multi-Way Merge Sort (M buffers)
Initial Runs	B (each run is 1 page)	$\lceil B/M \rceil$ (each run is M pages)
Merge Fan-in	2 (merges 2 runs at a time)	$M - 1$ (merges many runs at once)
Passes Formula	$\lceil \log_2 B \rceil$	$1 + \lceil \log_{M-1}(B/M) \rceil$

7 Page-Oriented Nested-Loop Joins

7.1 Task-by-Task Logic

Based on the formula for page-oriented nested-loop joins, follow these steps:

Identify the Page Counts (B)

- Relation r_1 : 100,000 tuples / 50 tuples per page = 2,000 pages
- Relation r_2 : 500,000 tuples / 10 tuples per page = 50,000 pages

Define the Roles

- **Outer Relation (R):** r_1 (2,000 pages)
- **Inner Relation (S):** r_2 (50,000 pages)

Apply the Cost Formula

- In a page-oriented nested-loop join, the outer relation is read exactly once.
- For every page in the outer relation, the entire inner relation must be scanned once.
- **Formula:** $Cost = \text{Pages}_{outer} + (\text{Pages}_{outer} \times \text{Pages}_{inner})$