

# DBS Exam Notes

January 12, 2026

## Contents

<b>1 Functional Dependencies</b>	<b>2</b>
1.1 The "Boss and Follower" Logic . . . . .	2
1.2 Verification Results . . . . .	2
<b>2 Finding Superkeys</b>	<b>2</b>
2.1 Step-by-Step Process . . . . .	2
<b>3 Normal Form Audit</b>	<b>3</b>
3.1 BCNF Check . . . . .	3
3.2 3NF Check . . . . .	3
<b>4 Lossless Join</b>	<b>3</b>
<b>5 External Merge Sort: 2-Way Algorithm</b>	<b>3</b>
5.1 Problem Example . . . . .	3
5.2 Step-by-Step Breakdown . . . . .	3
5.3 Why the Other Options Are Wrong . . . . .	4
<b>6 Multi-Way External Merge Sort</b>	<b>4</b>
6.1 Example: Using Relation $r_1$ . . . . .	4
6.2 Quick Comparison . . . . .	5
<b>7 Page-Oriented Nested-Loop Joins</b>	<b>5</b>
7.1 Task-by-Task Logic . . . . .	5
<b>8 Selection Size Estimation</b>	<b>5</b>
8.1 How to Calculate the Selections . . . . .	6
8.2 Verifying Multiple Options . . . . .	6
8.3 Summary Table . . . . .	6

# 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" ( $\alpha$ ), they **must** have the same value for the "Follower" ( $\beta$ ).

### 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:**  $\text{Cost} = \text{Pages}_{\text{outer}} + (\text{Pages}_{\text{outer}} \times \text{Pages}_{\text{inner}})$

## 8 Selection Size Estimation

When estimating the result size of selection queries, we need to understand several key parameters:

- $n_r$ : The total number of tuples (rows) in the relation  $r$ . For example,  $n_r = 4,000$ .

- $V(A, r)$ : The number of distinct values for attribute  $A$  in relation  $r$ . For instance,  $V(A, r) = 100$ ,  $V(B, r) = 200$ , and  $V(C, r) = 40$ .
- $[min, max]$ : The range of possible values for an attribute. For example, attribute  $B$  ranges from 20 to 60.

## 8.1 How to Calculate the Selections

### 1. Equality Selection ( $\sigma_{A=v}(r)$ )

**Logic:** If values are uniformly distributed, every distinct value is expected to appear the same number of times.

**Formula:** Estimated Size =  $\frac{n_r}{V(A, r)}$

**Example:**  $\sigma_{A=50}(r)$

- Calculation:  $4,000/100 = 40$
- Verdict: If a statement says the estimation is 100, it would be False.

### 2. Range Selection ( $\sigma_{A < v}(r)$ or $\sigma_{A > v}(r)$ )

**Logic:** You calculate the "fraction" of the range that is covered by the query and multiply it by the total number of tuples.

**Formula:** Estimated Size =  $n_r \times \frac{v - \min(A)}{\max(A) - \min(A)}$

**Example:**  $\sigma_{A < 50}(r)$  where range for  $A$  is  $[0, 100]$

- Calculation:  $4,000 \times \frac{50-0}{100-0} = 4,000 \times 0.5 = 2,000$

## 8.2 Verifying Multiple Options

### Option (a): $\sigma_{C=150}(r)$

- Calculation:  $n_r/V(C, r) = 4,000/40 = 100$
- Verdict: If a statement says the estimation is 100, it is True.

### Option (e): $\sigma_{B < 30}(r)$ where range for $B$ is $[20, 60]$

- Calculation:  $4,000 \times \frac{30-20}{60-20} = 4,000 \times \frac{10}{40} = 4,000 \times 0.25 = 1,000$
- Verdict: If a statement says the estimation is 1,000, it is True.

## 8.3 Summary Table

Type of Query	Formula
Equality ( $A = v$ )	$\frac{n_r}{V(A, r)}$
Range ( $A < v$ )	$n_r \times \frac{v - \min}{\max - \min}$
Range ( $A > v$ )	$n_r \times \frac{\max - v}{\max - \min}$