DBS Exam Notes

January 12, 2026

Contents

1 Functional Dependencies

1.1 The "Boss and Follower" Logic

A functional dependency $\alpha \to \beta$ 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

- $A \to B$ **(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.
- $A \to C$ **(Holds):**
  - For the duplicate boss **a2**, both rows have the identical follower **c2** in column C.
- $A \to D$ **(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 $B \to A$, 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 $AB \to C$, 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.