# Advanced Database Systems

Spring 2026

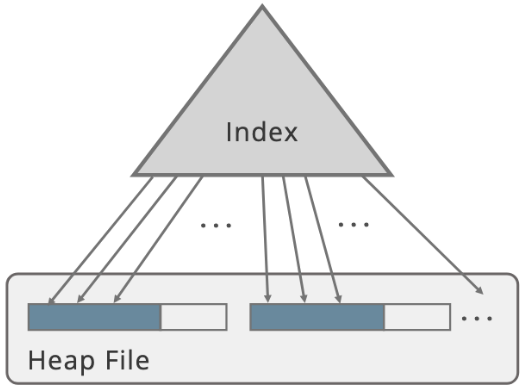
## Q&A Session 3

## About This Session

- Practice Worksheet 3 is now available on Learn
  - We will work through one question together during this session
- SQL Isolation Levels (non-examinable)
  - Why most database systems do **not** guarantee serializability by default
  - What weaker isolation levels mean in practice
  - Useful background for understanding real systems
- PostgreSQL Demonstration
  - `transaction_demo.sql` is available on Learn → Practice Worksheets
  - Run the script step by step and observe transaction behaviour

## Question 1

(Cost Estimation, Exam 2025) Consider a clustered B+ tree index with index entries stored using variant B, meaning that leaf nodes contain index entries in the form $(key, rid)$. The data records are stored in a heap file, which is initially sorted.The heap pages storing the data records are 2/3 full to accommodate future inserts.



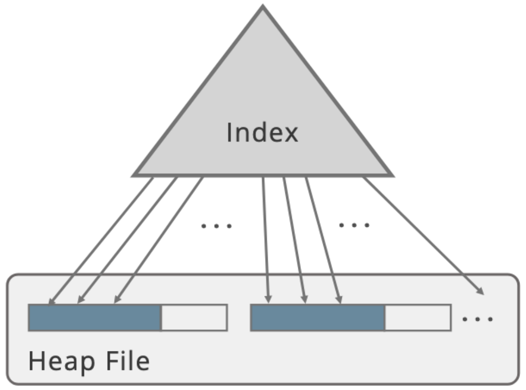
Consider the following notation:

- $P$ denotes the number of pages needed to store *compactly* all the data records (i.e., without leaving any free space).
- $R$ denotes the number of records that can fit on a page.
- $F$ denotes the average inner-node fanout in the index.
- $E$ denotes the average number of $(key, rid)$ pairs per leaf node in the index.
- For simplicity, assume that each leaf can accommodate $R$ such pairs – that is, the data records and the $(key, rid)$ pairs have the same size.

## Part (a)

## I/O cost of scanning all data records

(Cost Estimation, Exam 2025) Consider a clustered B+ tree index with index entries stored using variant B, meaning that leaf nodes contain index entries in the form $(key, rid)$. The data records are stored in a heap file, which is initially sorted.The heap pages storing the data records are 2/3 full to accommodate future inserts.



Consider the following notation:

- $P$ denotes the number of pages needed to store *compactly* all the data records (i.e., without leaving any free space).
- $R$ denotes the number of records that can fit on a page.
- $F$ denotes the average inner-node fanout in the index.
- $E$ denotes the average number of $(key, rid)$ pairs per leaf node in the index.
- For simplicity, assume that each leaf can accommodate $R$ such pairs – that is, the data records and the $(key, rid)$ pairs have the same size.

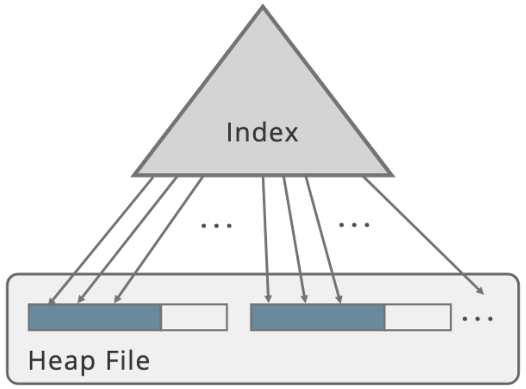
## Part (a)

## I/O cost of scanning all data records

- No need to use the index!
- Just scan the heap file
- How many heap pages exist?
  - P pages when stored compactly
  - But heap pages are 2/3 full

✅ **Cost = 3/2 · P**

(Cost Estimation, Exam 2025) Consider a clustered B+ tree index with index entries stored using variant B, meaning that leaf nodes contain index entries in the form $(key, rid)$. The data records are stored in a heap file, which is initially sorted.The heap pages storing the data records are 2/3 full to accommodate future inserts.



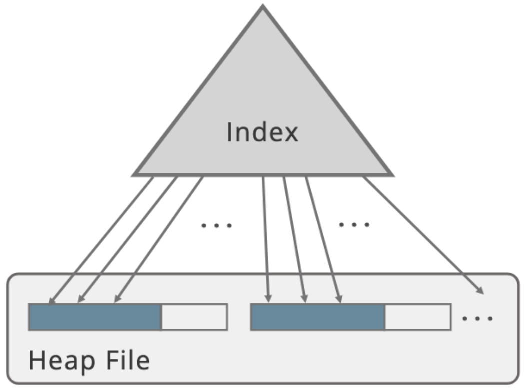
Consider the following notation:

- $P$ denotes the number of pages needed to store *compactly* all the data records (i.e., without leaving any free space).
- $R$ denotes the number of records that can fit on a page.
- $F$ denotes the average inner-node fanout in the index.
- $E$ denotes the average number of $(key, rid)$ pairs per leaf node in the index.
- For simplicity, assume that each leaf can accommodate $R$ such pairs – that is, the data records and the $(key, rid)$ pairs have the same size.

## Part (b)

## I/O cost of searching on a key attribute

(Cost Estimation, Exam 2025) Consider a clustered B+ tree index with index entries stored using variant B, meaning that leaf nodes contain index entries in the form $(key, rid)$. The data records are stored in a heap file, which is initially sorted.The heap pages storing the data records are 2/3 full to accommodate future inserts.



Consider the following notation:

- $P$ denotes the number of pages needed to store *compactly* all the data records (i.e., without leaving any free space).
- $R$ denotes the number of records that can fit on a page.
- $F$ denotes the average inner-node fanout in the index.
- $E$ denotes the average number of $(key, rid)$ pairs per leaf node in the index.
- For simplicity, assume that each leaf can accommodate $R$ such pairs – that is, the data records and the $(key, rid)$ pairs have the same size.

## Part (b)

## I/O cost of searching on a key attribute

Scanning heap file is inefficient, better use the index *(assumed the index is on the key attribute)*

How expensive is index search?

# of pages from root to matching leaf +

1 I/O to read matching data record

How many index entries of form *(key, rid)*?

$P \cdot R$

How many leaf index pages?

$P \cdot R / E$

What is the index height?

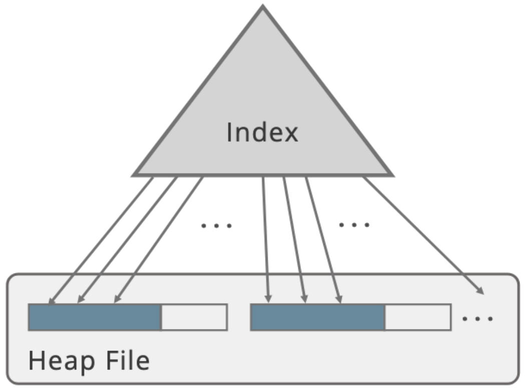
$\lceil \log_F (P \cdot R / E) \rceil$

Number of pages from root to matching leaf?

$\lceil \log_F (P \cdot R / E) \rceil + 1$

✅ **Cost =** $\lceil \log_F (P \cdot R / E) \rceil + 2$

(Cost Estimation, Exam 2025) Consider a clustered B+ tree index with index entries stored using variant B, meaning that leaf nodes contain index entries in the form $(key, rid)$. The data records are stored in a heap file, which is initially sorted.The heap pages storing the data records are 2/3 full to accommodate future inserts.



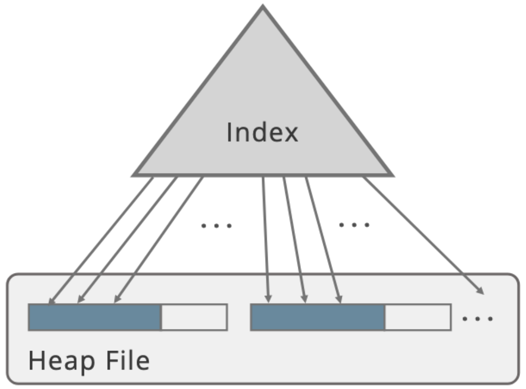
Consider the following notation:

- $P$ denotes the number of pages needed to store *compactly* all the data records (i.e., without leaving any free space).
- $R$ denotes the number of records that can fit on a page.
- $F$ denotes the average inner-node fanout in the index.
- $E$ denotes the average number of $(key, rid)$ pairs per leaf node in the index.
- For simplicity, assume that each leaf can accommodate $R$ such pairs – that is, the data records and the $(key, rid)$ pairs have the same size.

## Part (c)

## Range search returning M records

(Cost Estimation, Exam 2025) Consider a clustered B+ tree index with index entries stored using variant B, meaning that leaf nodes contain index entries in the form $(key, rid)$. The data records are stored in a heap file, which is initially sorted.The heap pages storing the data records are 2/3 full to accommodate future inserts.



Consider the following notation:

- $P$ denotes the number of pages needed to store *compactly* all the data records (i.e., without leaving any free space).
- $R$ denotes the number of records that can fit on a page.
- $F$ denotes the average inner-node fanout in the index.
- $E$ denotes the average number of $(key, rid)$ pairs per leaf node in the index.
- For simplicity, assume that each leaf can accommodate $R$ such pairs – that is, the data records and the $(key, rid)$ pairs have the same size.

## Part (c)

## Range search returning M records

To reach leaf page but not read it

$[ \log_F (P \cdot R / E) ]$

How many leaf index entries to read?

M

*(strictly speaking, M+1 as you need to detect the end of range)*

How many leaf index pages to read?

$[ M / E ]$

How many data records to read from heap file?

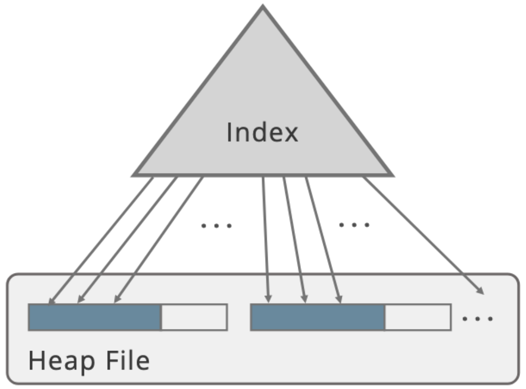
M

How many pages to read from heap file?

$[ 3 / 2 \cdot M / R ]$

✓ **Cost =** $[ \log_F (P \cdot R / E) ] +$
$[ M / E ] +$
$[ 3 / 2 \cdot M / R ]$

(Cost Estimation, Exam 2025) Consider a clustered B+ tree index with index entries stored using variant B, meaning that leaf nodes contain index entries in the form $(key, rid)$. The data records are stored in a heap file, which is initially sorted.The heap pages storing the data records are 2/3 full to accommodate future inserts.



Consider the following notation:

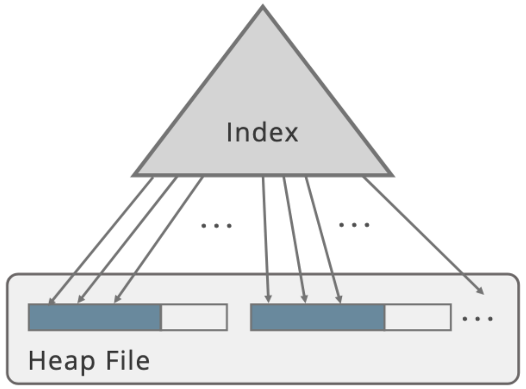
- $P$ denotes the number of pages needed to store *compactly* all the data records (i.e., without leaving any free space).
- $R$ denotes the number of records that can fit on a page.
- $F$ denotes the average inner-node fanout in the index.
- $E$ denotes the average number of $(key, rid)$ pairs per leaf node in the index.
- For simplicity, assume that each leaf can accommodate $R$ such pairs – that is, the data records and the $(key, rid)$ pairs have the same size.

## Part (d)

## Inserting or deleting one record

Assuming no changes in tree structure

(Cost Estimation, Exam 2025) Consider a clustered B+ tree index with index entries stored using variant B, meaning that leaf nodes contain index entries in the form $(key, rid)$. The data records are stored in a heap file, which is initially sorted.The heap pages storing the data records are 2/3 full to accommodate future inserts.



Consider the following notation:

- $P$ denotes the number of pages needed to store *compactly* all the data records (i.e., without leaving any free space).
- $R$ denotes the number of records that can fit on a page.
- $F$ denotes the average inner-node fanout in the index.
- $E$ denotes the average number of $(key, rid)$ pairs per leaf node in the index.
- For simplicity, assume that each leaf can accommodate $R$ such pairs – that is, the data records and the $(key, rid)$ pairs have the same size.

## Part (d)

## Inserting or deleting one record

Assuming no changes in tree structure

Search cost: $\lceil \log_F (P \cdot R / E) \rceil$

Read leaf page → change it → write it back

Read heap page → change it → write it back

✅ **Cost =** $\lceil \log_F (P \cdot R / E) \rceil + 4$

## Isolation Levels in Practice

# Quick Recap: The 'I' in ACID

## Why isolation matters

- Multiple transactions run concurrently
- Isolation controls what effects of other transactions you can see
- Without isolation, concurrent execution can produce incorrect results

## Common anomalies

| Anomaly | Description |
|---|---|
| **Dirty Read** | T1 reads uncommitted data written by T2, which later rolls back |
| **Non-repeatable Read** | T1 reads the same row twice; T2 commits an update in between |
| **Phantom Read** | T1 re-executes a range query; T2 inserts/deletes rows in between |

# SQL Isolation Levels

SQL-92 standard defines four isolation levels in terms of anomalies

| Isolation Level | Dirty Read | Non-Repeatable Read | Phantom Read |
|---|---|---|---|
| `READ UNCOMMITTED` | Possible | Possible | Possible |
| `READ COMMITTED` | Not Possible | Possible | Possible |
| `REPEATABLE READ` | Not Possible | Not Possible | Possible |
| `SERIALIZABLE` | Not Possible | Not Possible | Not Possible |

**Key idea:** Higher isolation → fewer anomalies → less concurrency

# Isolation Levels in Real Systems

Most DBMS do <u>not</u> default to `SERIALIZABLE`

- Isolation level can be set per transaction

Why not `SERIALIZABLE` by default?

- **Serializable isolation is expensive:**
- More locking, lower throughput, more transaction aborts

What most systems choose

- Better performance + acceptable anomalies

Takeaways

- Isolation level choice is a systems trade-off among consistency, performance, and scalability
- Applications running on weaker isolation levels must explicitly handle possible anomalies

| DBMS | Default Isolation Level |
|---|---|
| PostgreSQL | `READ COMMITTED` |
| MySQL | `REPEATABLE READ` |
| Oracle | `REPEATABLE READ` |
| SQL Server | `READ COMMITTED` |
| SQLite | `SERIALIZABLE` |

## Choosing an Isolation Level in Practice

Start with the default (`READ COMMITTED`)

- Correct for the vast majority of OLTP workloads. Most bugs are logic errors, not isolation anomalies

Use `REPEATABLE READ` for multi-step reads

- Financial summaries, audit snapshots, batch reports – anywhere two reads of the same data must agree

Use `SERIALIZABLE` when your transaction reads data to decide what to write

- Serializable is the safe default for anything involving cross-row business rules
- E.g.: "only book a seat if it's free", "only approve a transfer if the balance is sufficient"

Never set `READ UNCOMMITTED` unless you know why

- Acceptable only for approximate analytics. Dirty reads can silently corrupt business logic

# PostgreSQL Demonstration

Script: `transaction_demo.sql`

- Available on Learn → Practice Worksheets

Open **two sessions** and run the script **step by step**

Observe:

- transaction behaviour
- blocking and locks
- deadlocks