Shrinking Segments

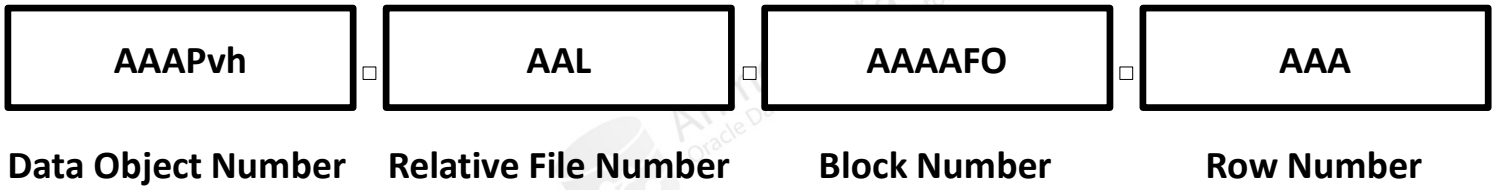
By Ahmed Baraka

Objectives

By the end of this lecture, you should be able to perform the following:

- Describe ROWID pseudocolumn
- Describe segment defragmentation
- Shrink database segments
- Deallocate unused space
- Move a table from its current location

ROWID Format



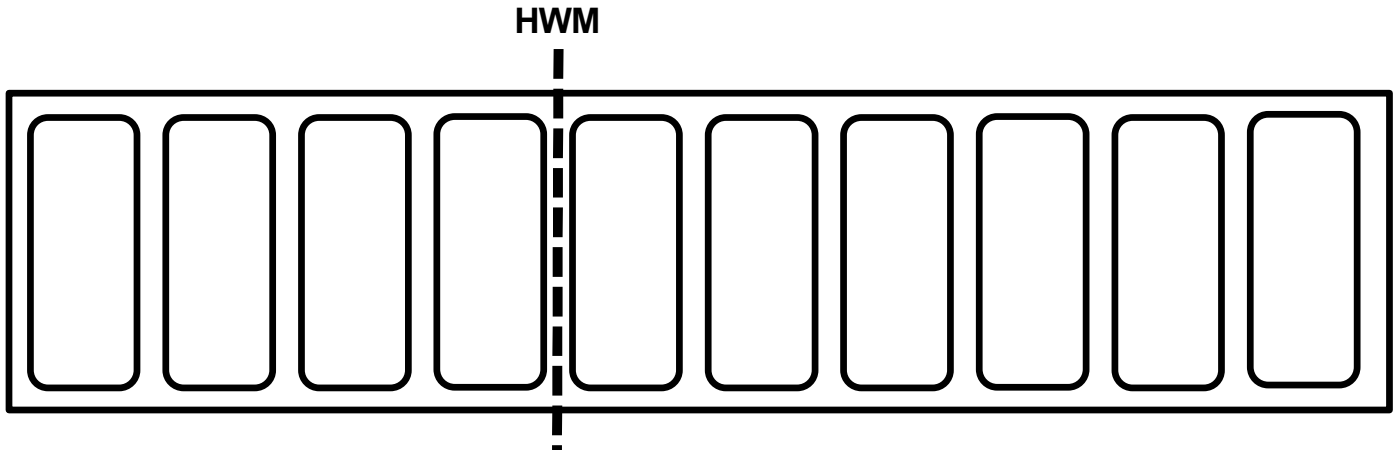
About ROWID Pseudocolumn

- Oracle Database internally uses a ROWID to uniquely identify a row.
- It holds the information enough to directly reach to the row in its location (data file/block number/row number)
- Once constructed for a row, it is not changed unless the segment is re-organized, shrunk, moved to a different location, or rebuilt.
- Indexes use ROWIDs to reference the rows in its table
- Can be queried in any table:

```
SELECT ROWID FROM employees WHERE employee_id = 100;
ROWID
------------------
AAAPvhAALAAAAFOAAA

SELECT employee_no FROM employees WHERE ROWID='AAAPvhAALAAAAFOAAA';
```

Segment Defragmentation

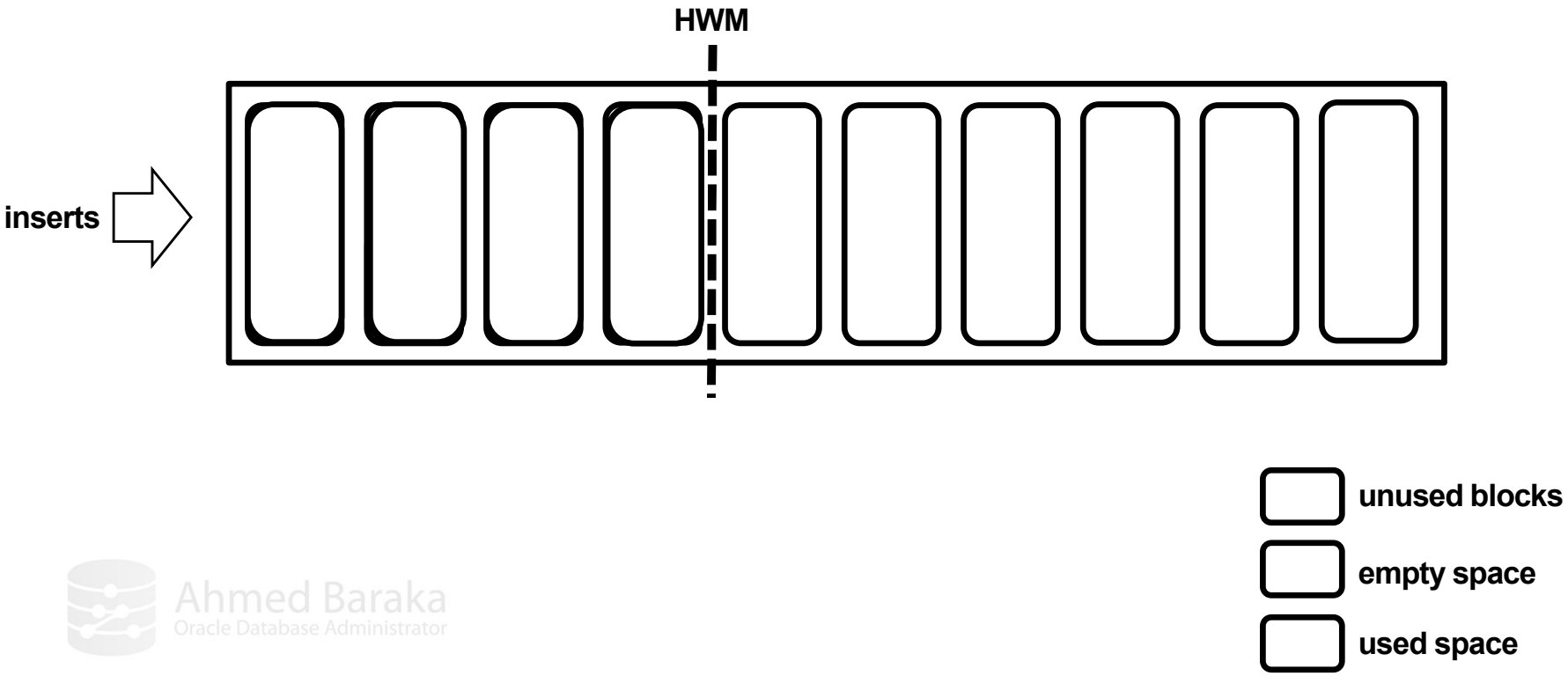


unused blocks

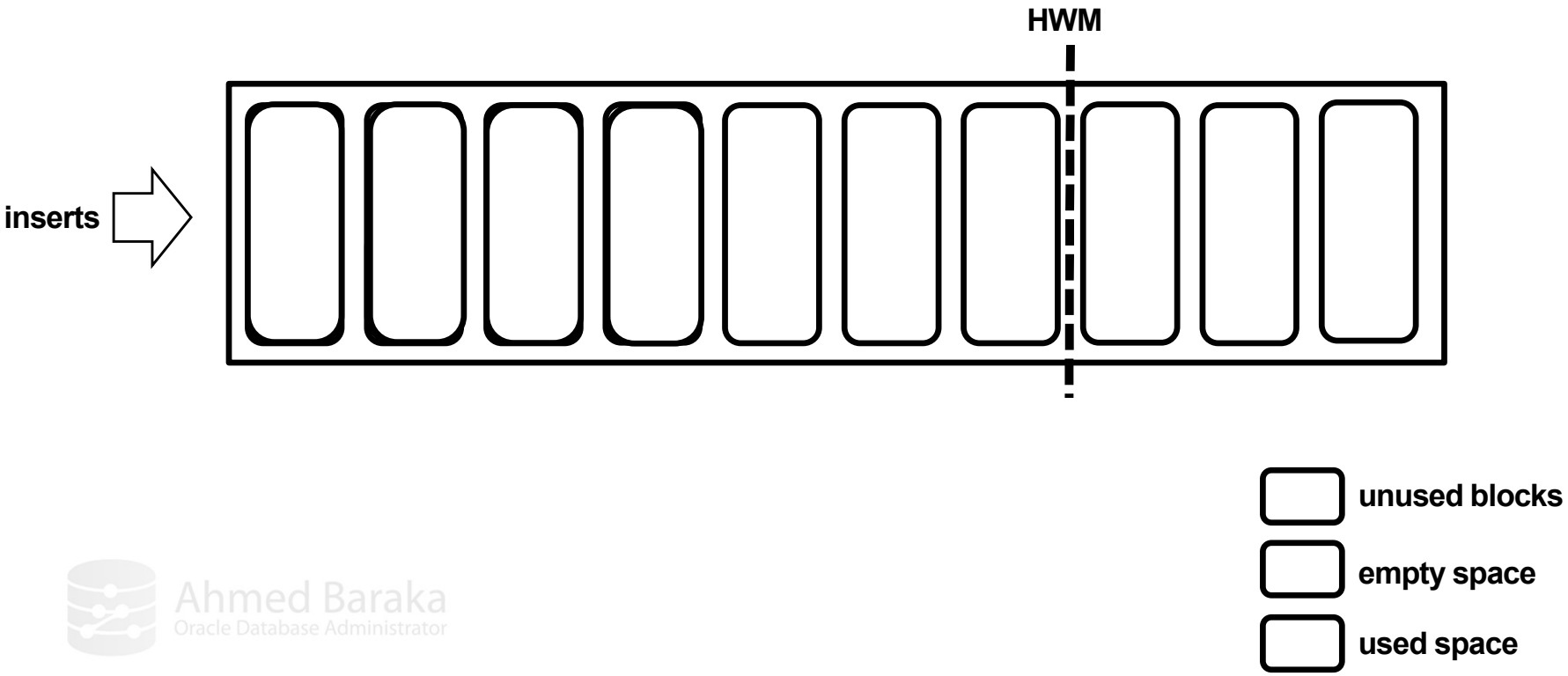
empty space

used space

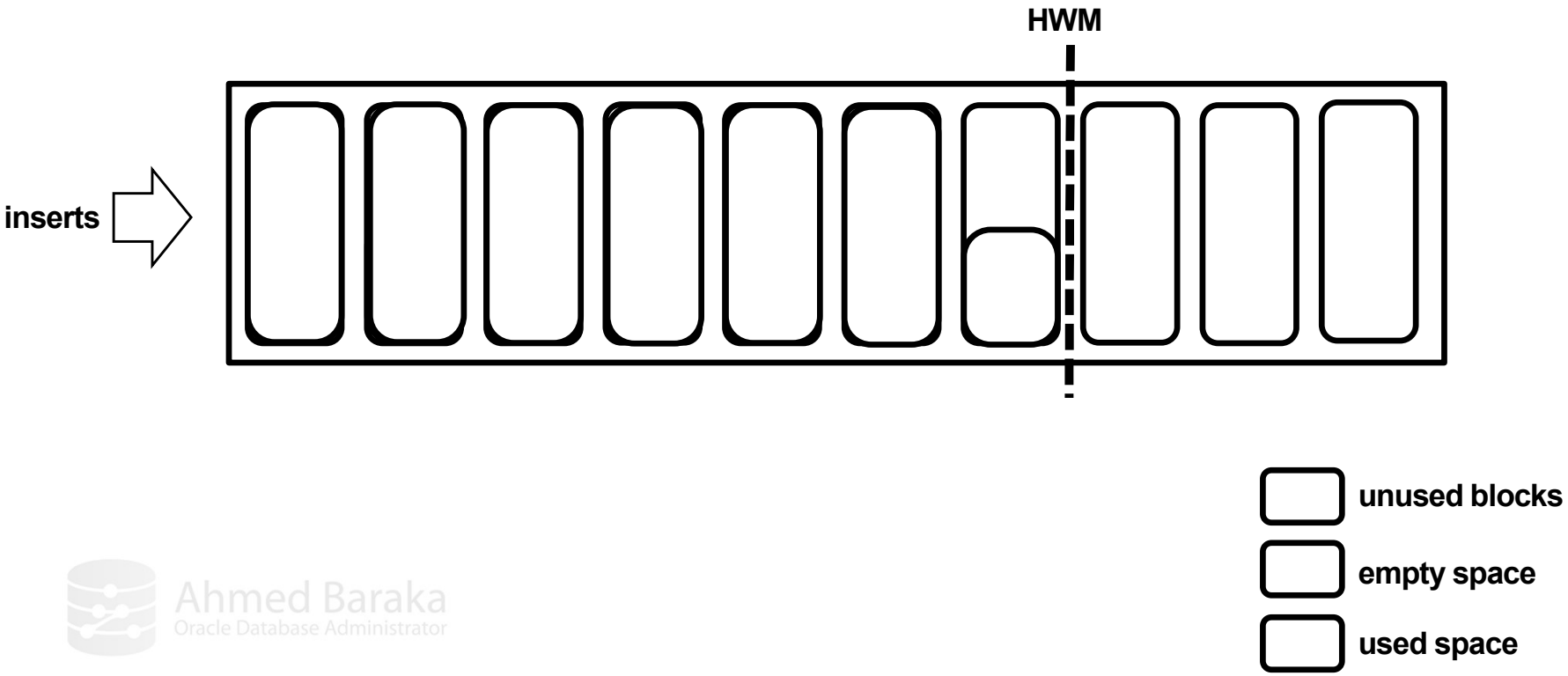
Segment Defragmentation



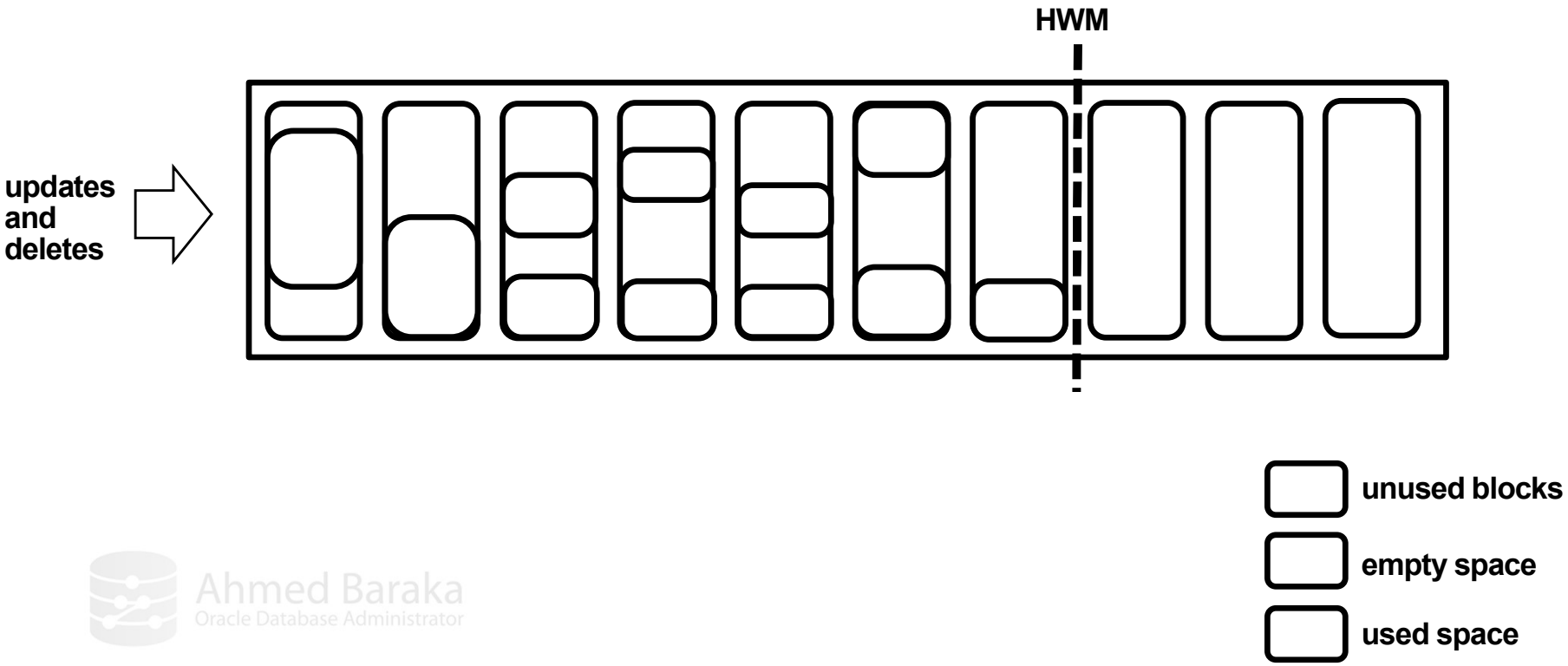
Segment Defragmentation



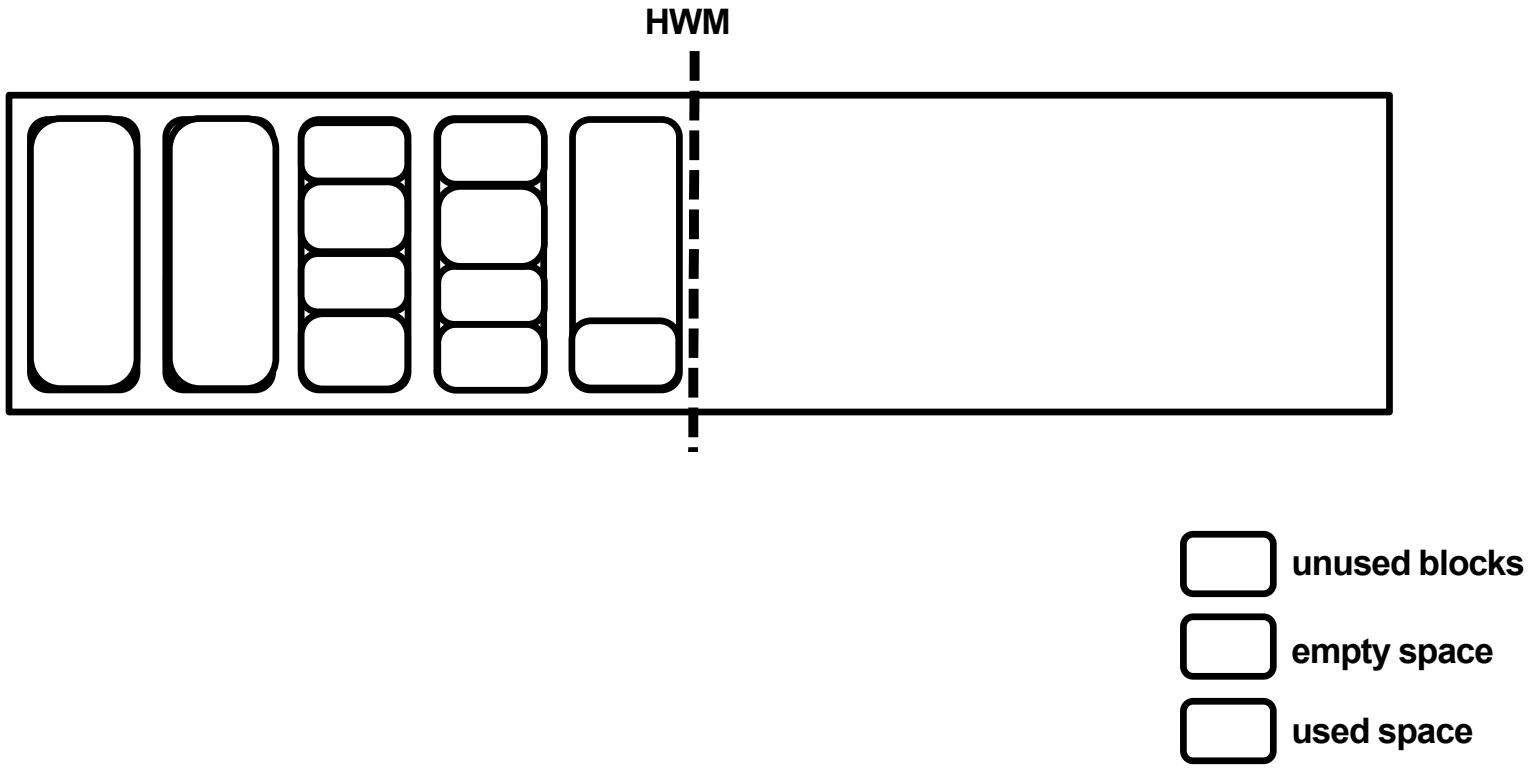
Segment Defragmentation



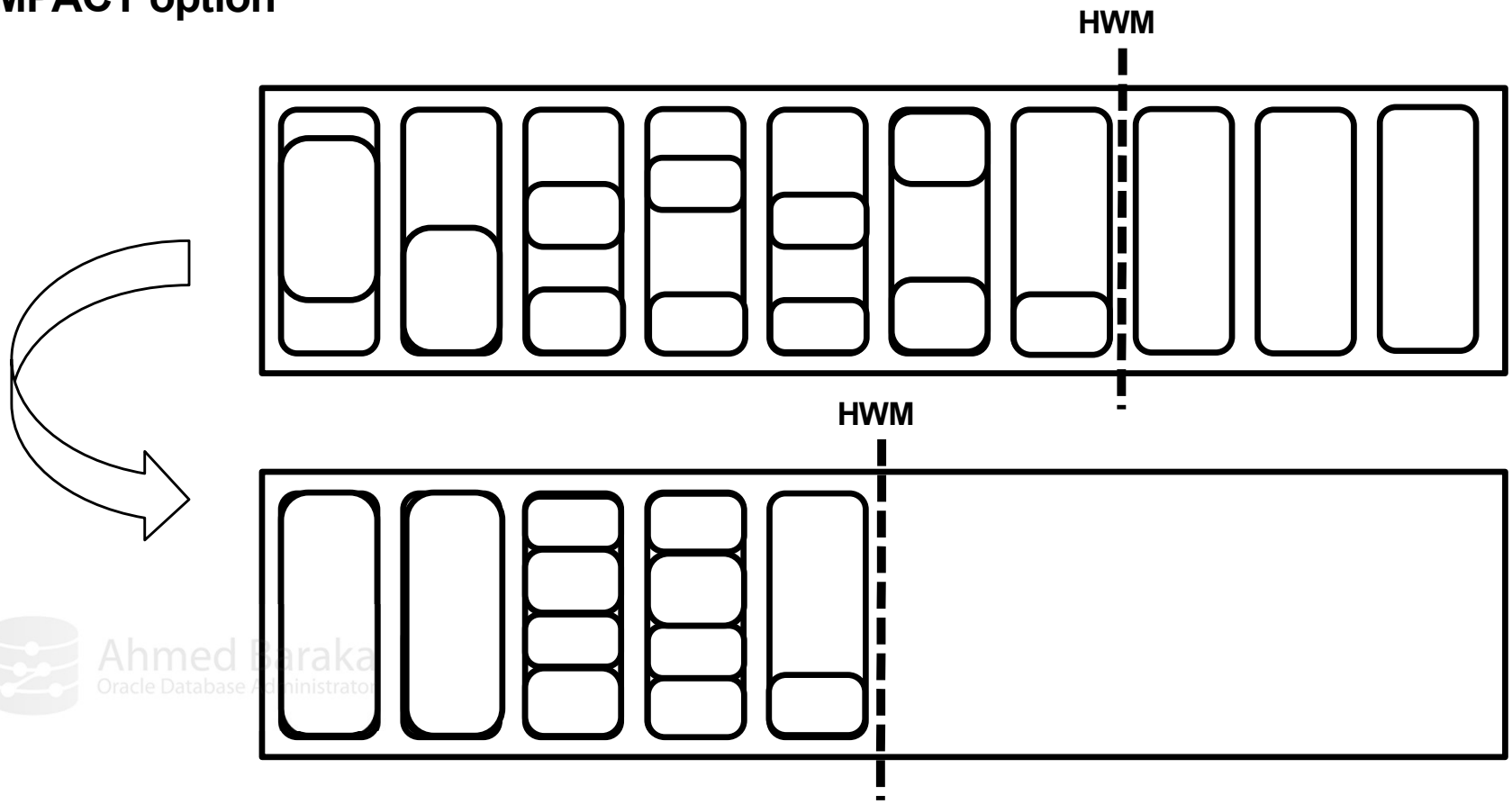
Segment Defragmentation



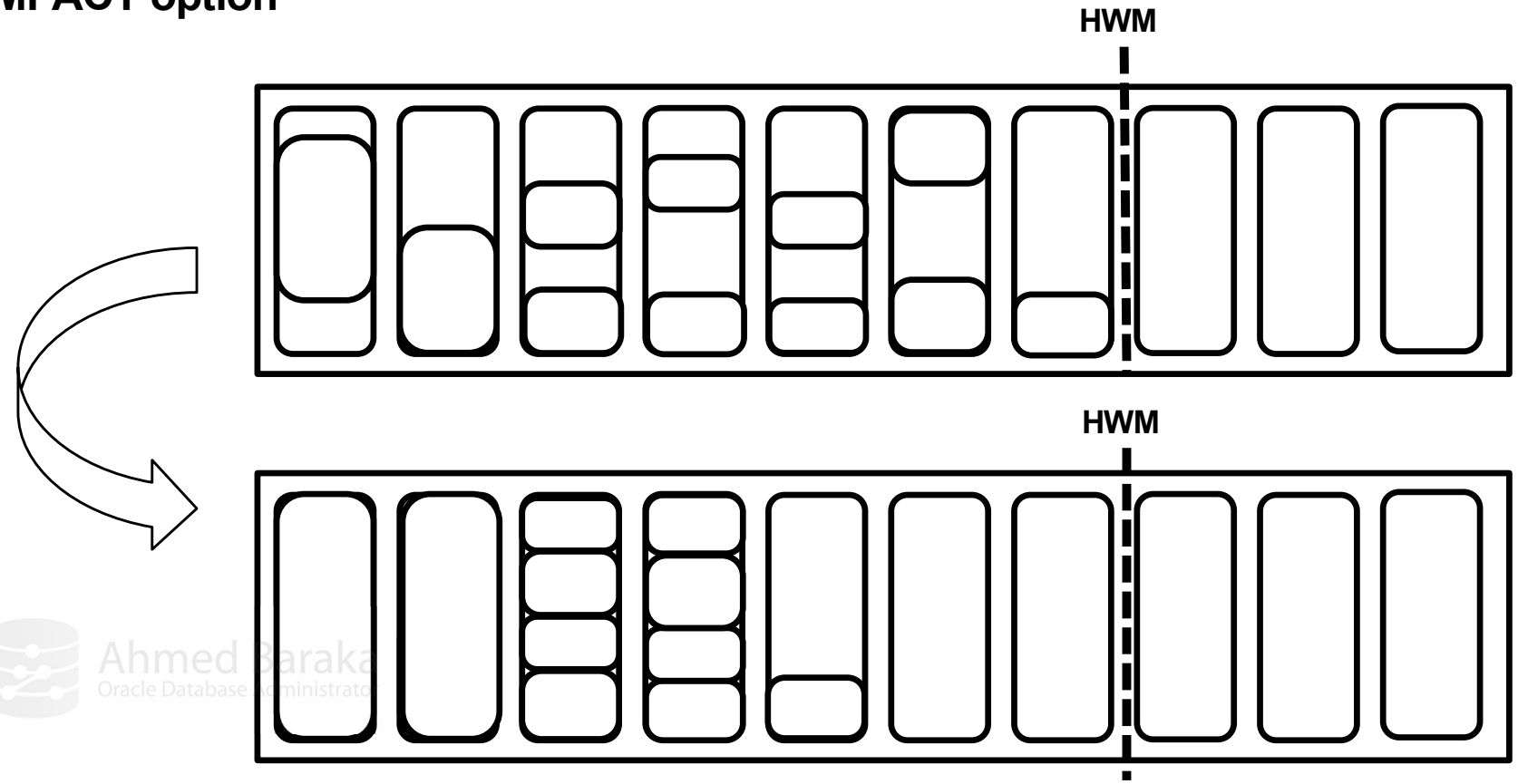
Segment Defragmentation



Shrinking Segments with no COMPACT option



Shrinking Segments with COMPACT option



Deallocating Unused Space

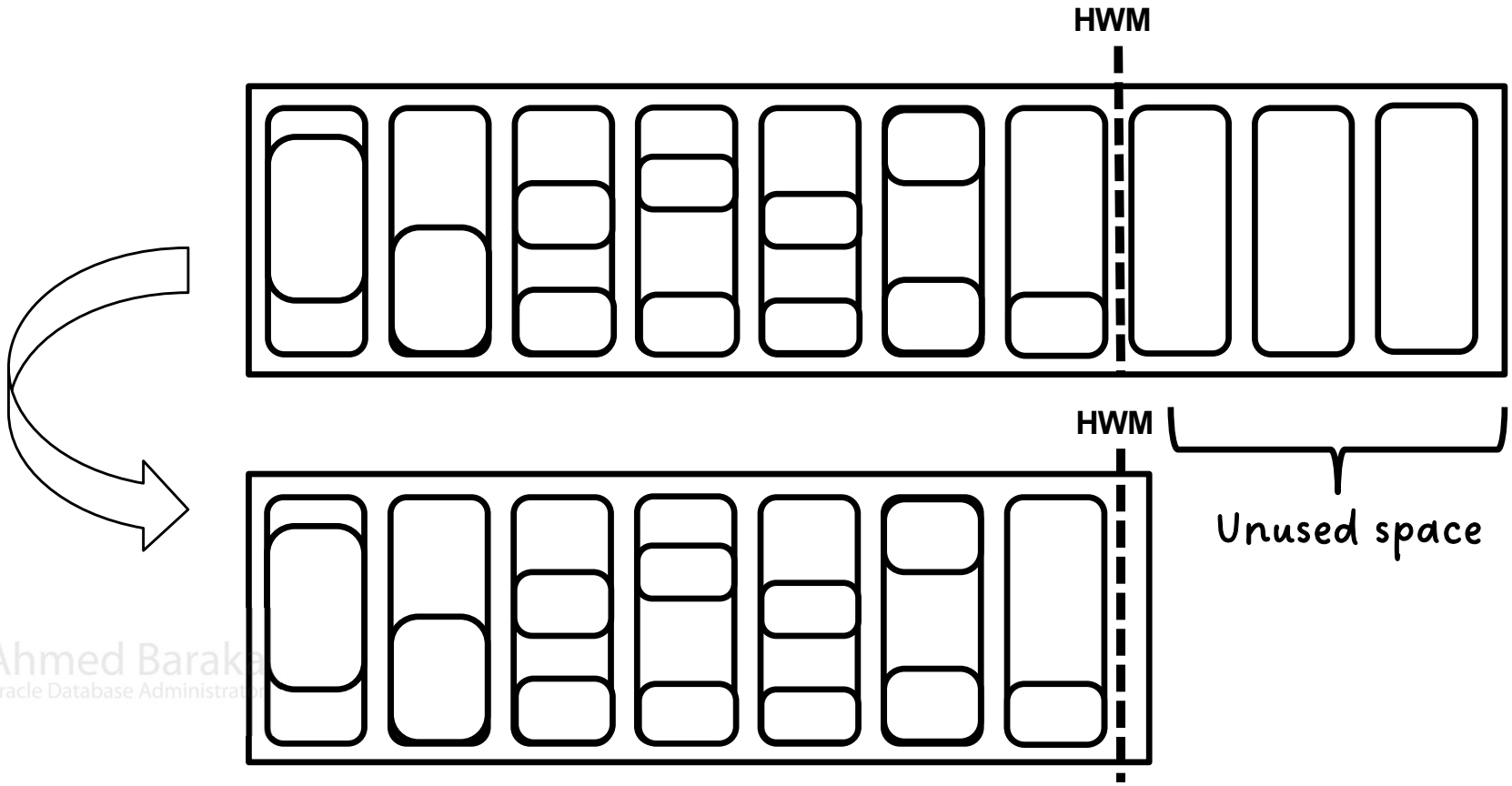


Table Defragmentation Solutions

- Shrinking tables
- Deallocating unused space
- Moving tables
- Online redefinition
- Using Data Pump Export/Import Utilities

About Shrinking Database Segments

- Default actions: compact data blocks, reset HWM, deallocate unused space
- Benefits: Better cache and space utilization, better performance
- Segment shrink is an online, in-place operation
- Can be applied on:
  - Tables
  - Partitions and sub-partitions
  - LOB segments
  - Materialized view logs
  - Index
  - Index-organized tables
  - Materialized views
- Valid only for segments in tablespaces with automatic segment management

Shrinking Database Segments

1. Enable row movement:

```
ALTER TABLE mytable ENABLE ROW MOVEMENT;
```

   - ROWIDs of the rows are changed
   - Indexes will be unusable
   - ROWID-based triggers should be disabled before issuing this statement
   - The statement fails if the table is being updated

2. Shrink the segment:

```
ALTER TABLE <tname> SHRINK SPACE [COMPACT] [CASCADE];
```

   - **COMPACT**: blocks are compacted but the HWM is not reset, i.e. space is not released. Useful when the object is under concurrent DML operations
   - **CASCADE**: shrink dependent objects

Measuring Deleted Space in Tables

- Deleted space can be measured:

```
ANALYZE TABLE my_table COMPUTE STATISTICS;

SELECT BLOCKS,
       BLOCKS*8192/1024 TOTAL_SIZE_KB,
       AVG_SPACE,
       round(BLOCKS*AVG_SPACE/1024,2) FREE_SPACE_KB
FROM USER_TABLES
WHERE TABLE_NAME='MY_TABLE';
```

Shrinking Tables Best Practice

- Mostly beneficial for FTS statements
- Consider it when high percentage (>20%) of the table is deleted space
- Use COMAPCT during peak hours and when concurrent DML operations are applied on the segment
- For indexes, rebuilding indexes usually provides better results thank shrinking:

```
ALTER INDEX CUST_NAME_IDX REBUILD;
```

About Deallocating Unused Space

- The database frees the **unused space** at the unused (on the top of the highwater mark) end of the database segment and makes it available for other segments
- Normally, shrinking is more useful.
- To apply it on a table:

```
ALTER TABLE <table> DEALLOCATE UNUSED [KEEP integer];
```

Moving a Table

- When a table is moved to a new location (within the same tablespace or to another tablespace), it gets defragmented
- Row movement must be enabled
- To move a table:

```
ALTER TABLE <t-name> MOVE [ONLINE] TABLESPACE <tbs-name> [UPDATE
INDEXES];
```

Summary

In this lecture, you should have learnt how to perform the following:

- Describe ROWID pseudocolumn
- Describe segment defragmentation
- Shrink database segments
- Deallocate unused space
- Move a table from its current location