

Organization of Records in Blocks

Read Sec. 4.2 Riguzzi et al. Sistemi Informativi

Slides derived from those by Hector Garcia-Molina

Topic

- How to lay out records on blocks

To represent:

- Integer (short): 2 bytes
e.g., 35 is

00000000	00100011
----------	----------

- Real, floating point
 n bits for mantissa, m for exponent....

To represent:

- Characters
 - various coding schemes suggested,
most popular is ascii

Example:

A: 1000001

a: 1100001

5: 0110101

LF: 0001010

To represent:

- Boolean

e.g., TRUE

1111 1111

FALSE

0000 0000

- Application specific

e.g., RED → 1 GREEN → 3

BLUE → 2 YELLOW → 4 ...

⇒ Can we use less than 1 byte/code?

Yes, but only if desperate...

To represent:

- Dates

- e.g.:
- Integer, # days since Jan 1, 1900
 - 8 characters, YYYYMMDD
 - 7 characters, YYYYDDD
- (not YYMMDD! Why?)

- Time

- e.g.
- Integer, seconds since midnight
 - characters, HHMMSSFF

To represent:

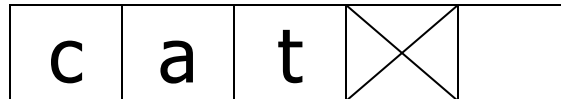
- Fixed length characters strings (CHAR(n)):
 - n bytes
 - If the value is shorter, fill the array with a *pad* character, whose 8-bit code is not one of the legal characters for SQL strings

c	a	t	x	x	x
---	---	---	---	---	---

To represent:

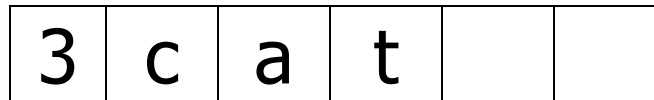
- Variable-length characters strings
(CHAR VARYING(n)): n+1 bytes max
 - Null terminated

e.g.,



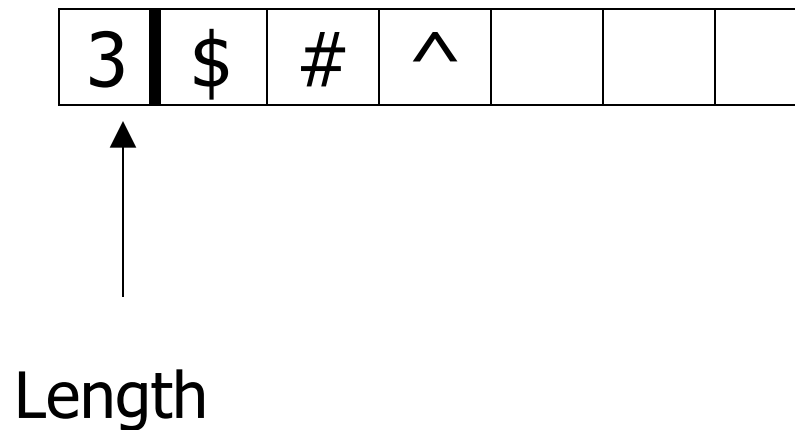
- Length given

e.g.,



To represent:

- BINARY VARYING(n)



Key Point

- Fixed length items
- Variable length items
 - usually length given at beginning

Overview

Data Items



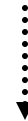
Records



Blocks



Files



Memory

Types of records:

- Main choices:
 - FIXED vs VARIABLE LENGTH

A SCHEMA (not record) contains
following information

- # fields
- type of each field
- order in record
- name of each field

Example: fixed length

Employee record

- (1) E#, 2 byte integer
- (2) E.name, 10 char.
- (3) Dept, 2 byte code

Schema

55	s m i t h	02
----	-----------	----

83	j o n e s	01
----	-----------	----

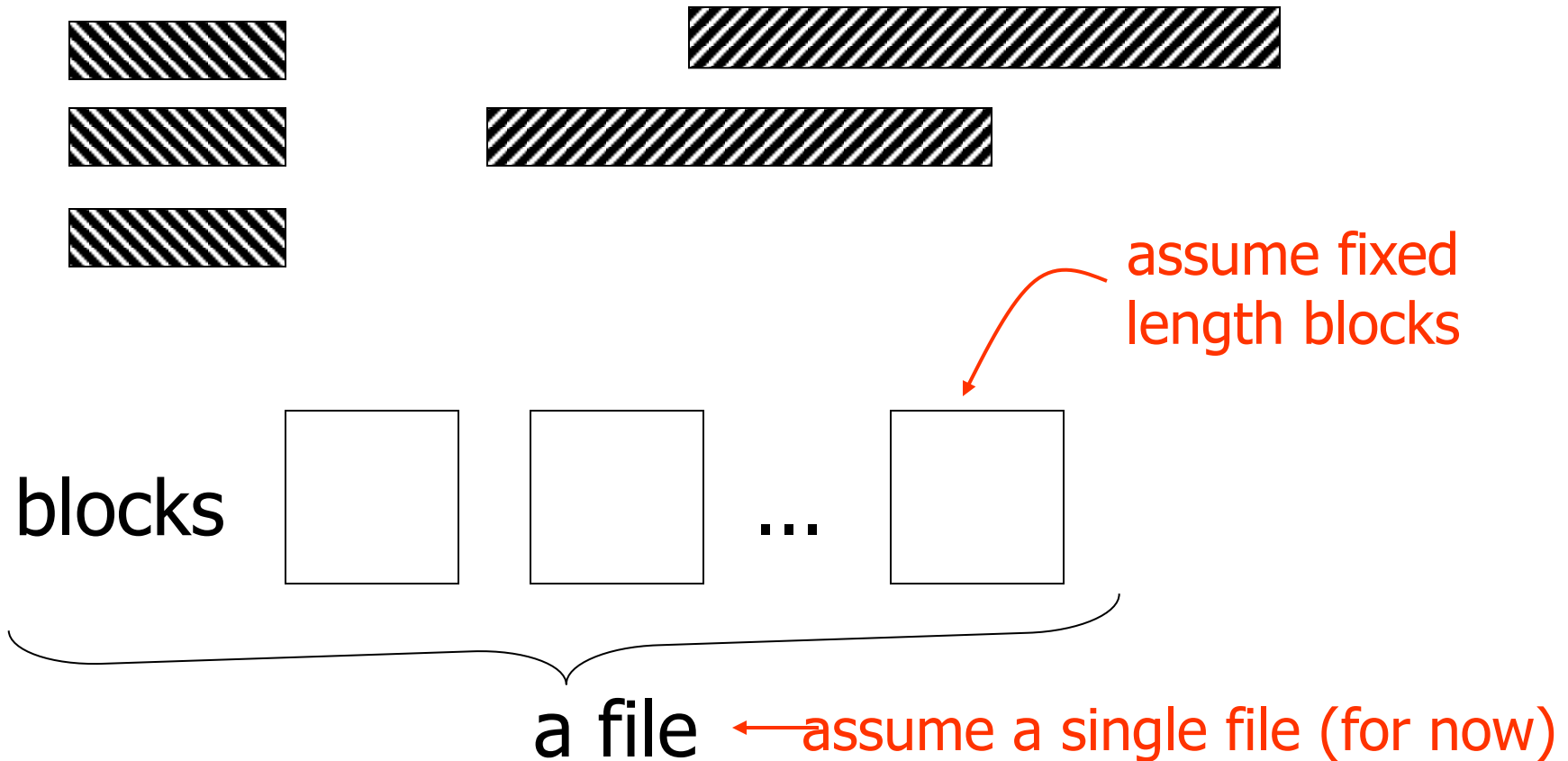
Records

Record header - data at beginning
that describes record

May contain:

- record type
- record length
- time stamp
- ...

Next: placing records into blocks



Options for storing records in blocks:

- (1) separating records
- (2) spanned vs. unspanned
- (3) mixed record types – clustering
- (4) split records
- (5) indirection

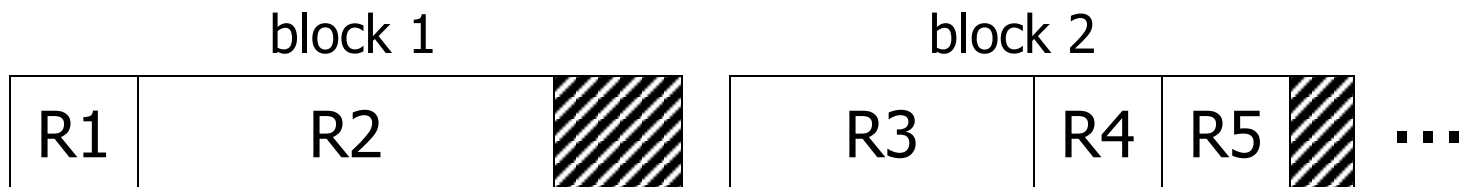
(1) Separating records



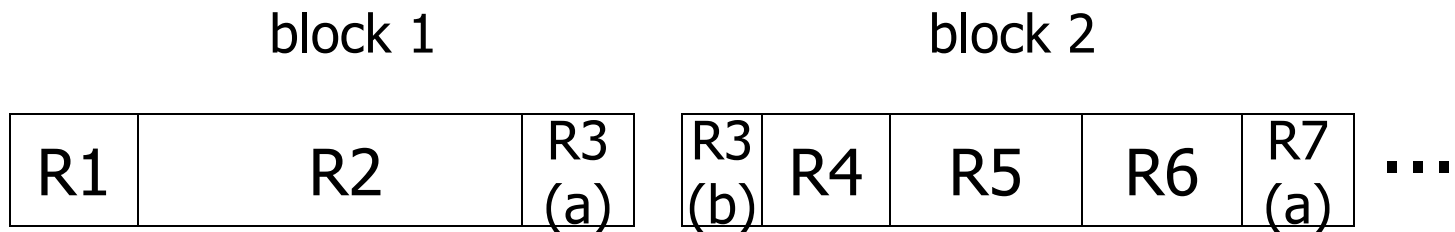
- (a) no need to separate - fixed size recs.
- (b) special marker
- (c) give record lengths (or offsets)
 - within each record
 - in block header

(2) Spanned vs. Unspanned

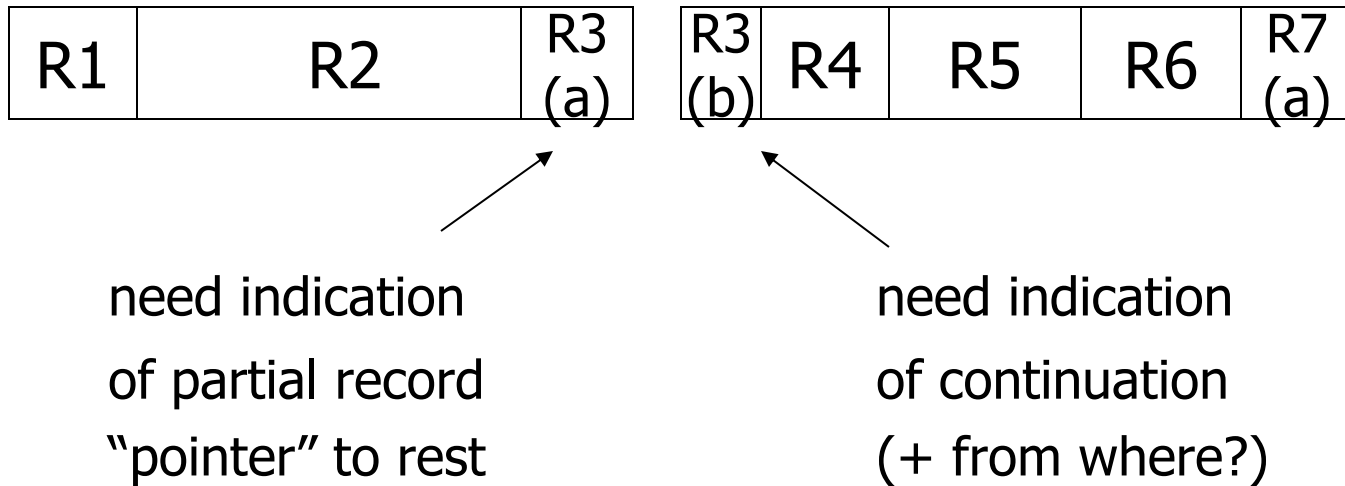
- Unspanned: records must be within one block



- Spanned



With spanned records:



Spanned vs. unspanned:

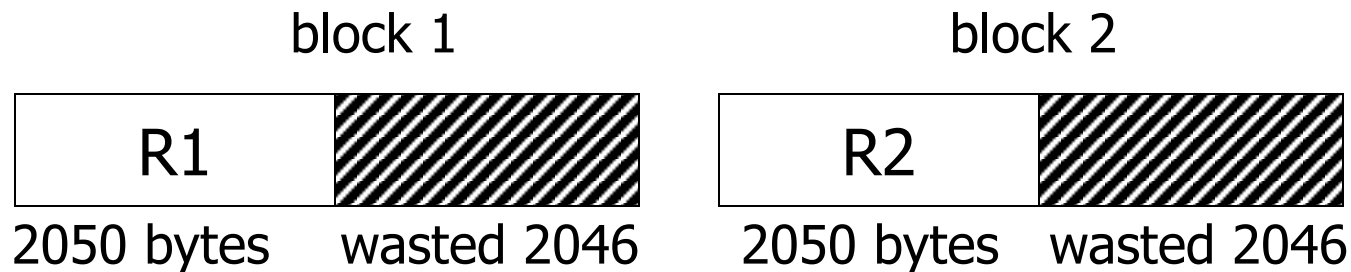
- Unspanned is much simpler, but may waste space...
- Spanned essential if
record size > block size

Example

10^6 records

each of size 2,050 bytes (fixed)

block size = 4096 bytes



- Total wasted = 2×10^9 Utiliz = 50%
- Total space = 4×10^9

(3) Mixed record types

- Mixed - records of different types
(e.g. EMPLOYEE, DEPT)
allowed in same block

e.g., a block

EMP	e1	DEPT	d1	DEPT	d2	
-----	----	------	----	------	----	--

Why do we want to mix?

Records that are frequently accessed together should be in the same block

CLUSTERING

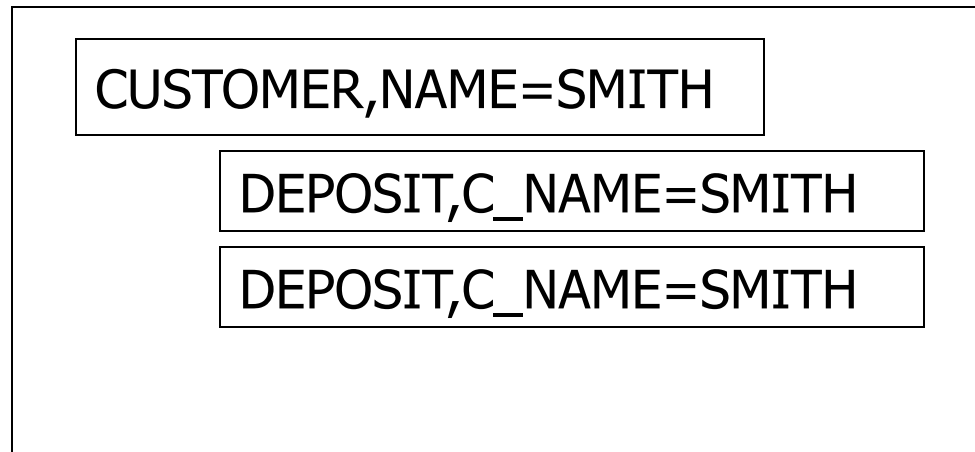
Compromise:

No mixing, but keep related records in same cylinder ...

Example

Q1: select A#, C_NAME, C_CITY, ...
from DEPOSIT, CUSTOMER
where DEPOSIT.C_NAME =
CUSTOMER.NAME

a block

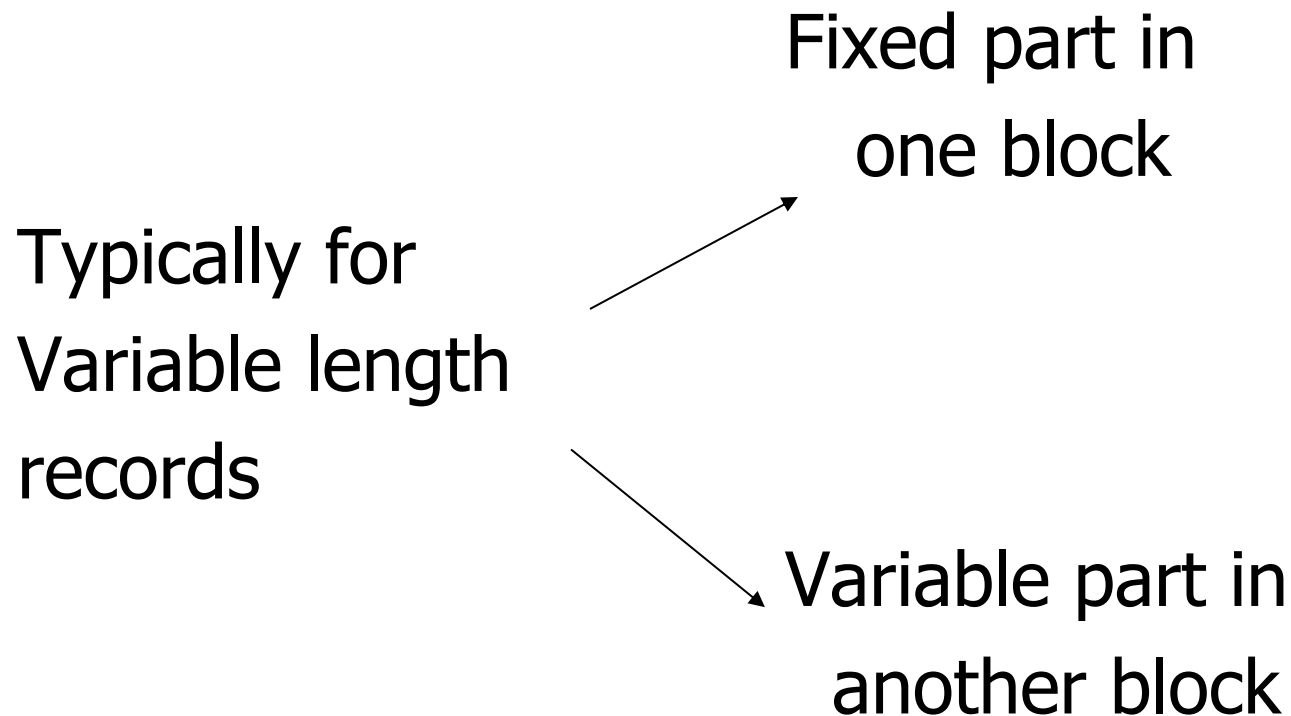


- If Q1 frequent, clustering good
- But if Q2 frequent

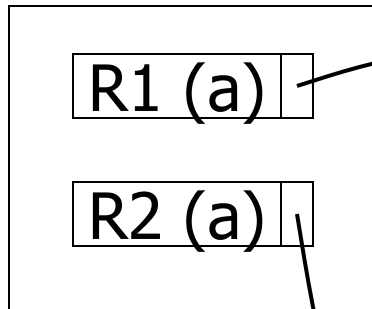
Q2: SELECT *
 FROM CUSTOMER

CLUSTERING IS COUNTER PRODUCTIVE

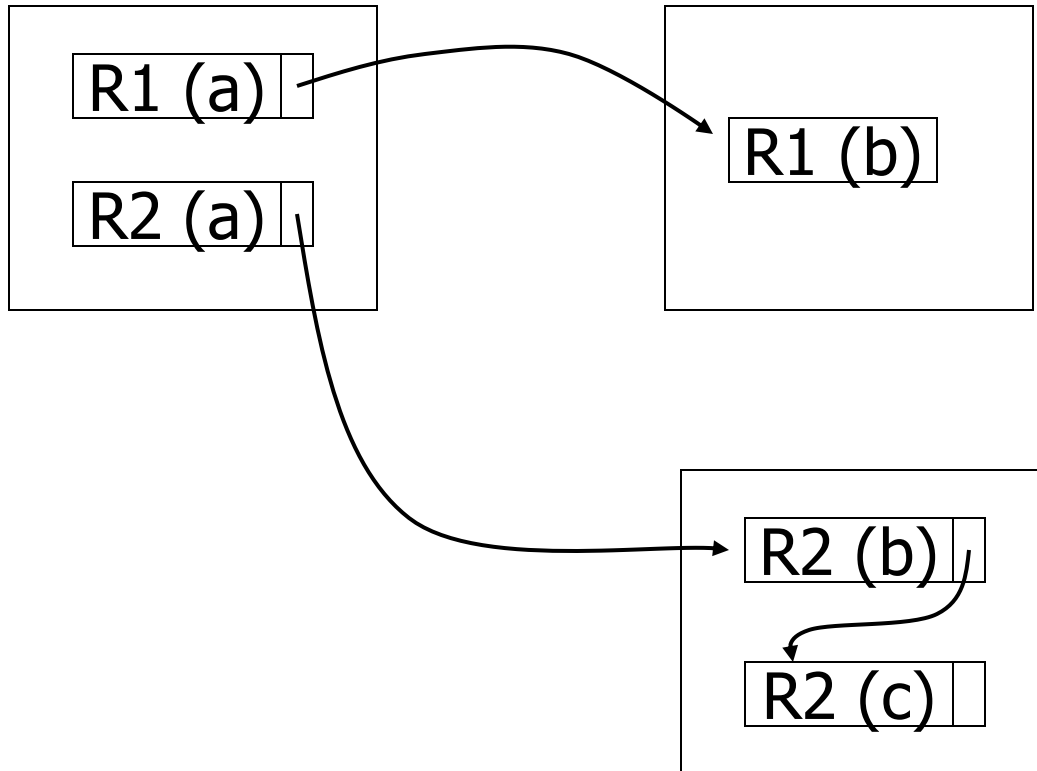
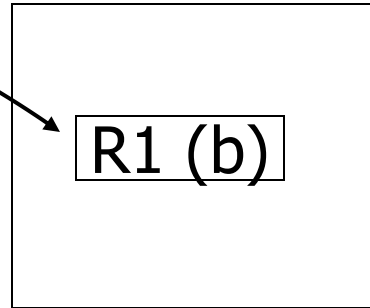
(4) Split records



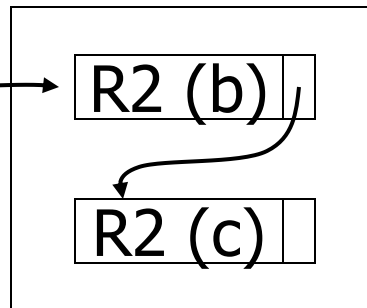
Block with fixed parts



Block with variable parts

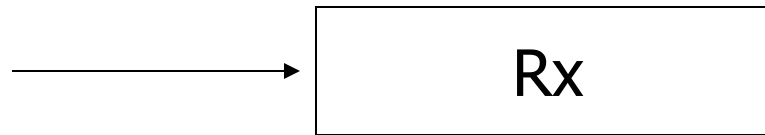


Block with
variable
parts



(5) Indirection

- How does one refer to records?



Many options:

Physical



Indirect

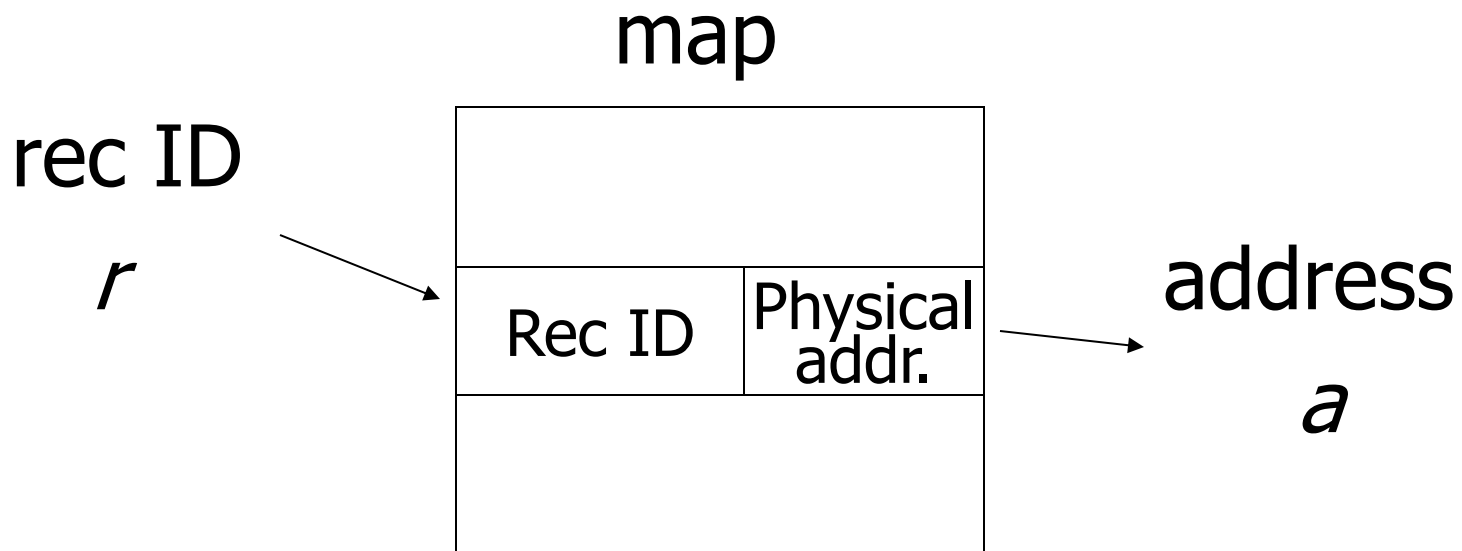
☆ Purely Physical

E.g., Record
Address
or ID = { Device ID
Cylinder #
Track #
Block #
Offset in block

Block ID

☆ Fully Indirect

E.g., Record ID is arbitrary bit string

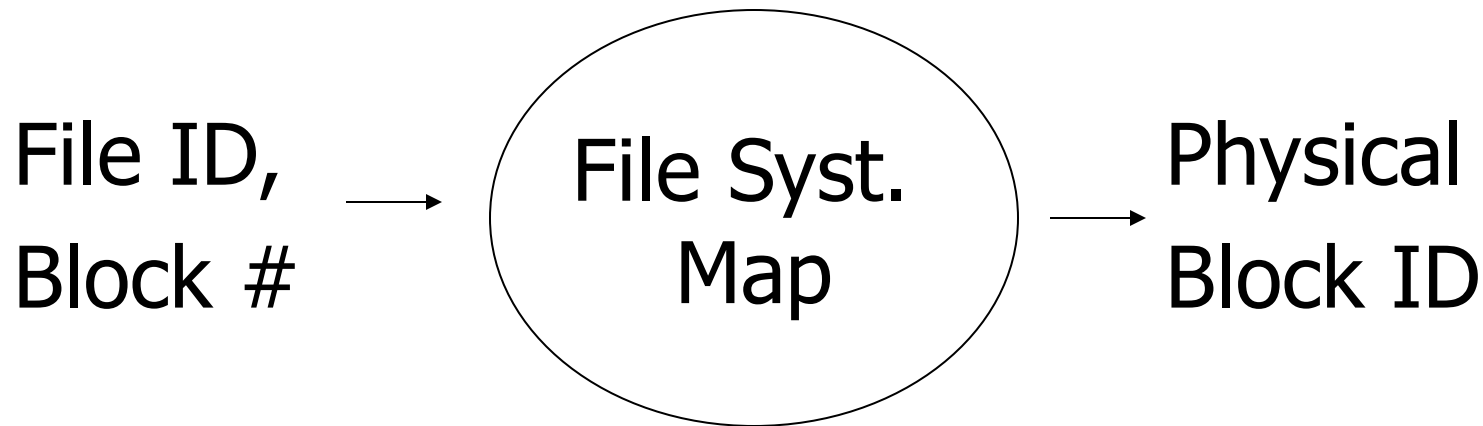


Typical Use logical block #'s understood by file system

→ File ID

Block #

Offset in block



Block header - data at beginning that describes block

May contain:

- File ID (or RELATION or DB ID)
- This block ID
- Record directory
- Pointer to free space
- Type of block (e.g. contains recs type 4;
is overflow, ...)
- Pointer to other blocks “like it”
- Timestamp ...

Other Topic

Insertion/Deletion

Options for deletion:

- (a) Immediately reclaim space
- (b) Mark deleted
 - May need chain of deleted records
(for re-use)
 - Need a way to mark:
 - special characters
 - delete field
 - in map

☆ As usual, many tradeoffs...

- How expensive is to move valid records to free space for immediate reclaim?
- How much space is wasted?
 - delete fields, free space chains,...