CS411
Database Systems

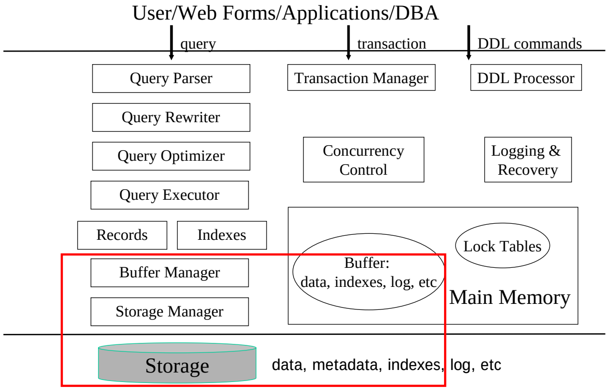
09: Storage

Kazuhiro Minami

CS411: Two Perspectives on DBMS

- User perspective
  - how to use a database system
    - Database design
    - Database programming
- System perspective
  - how to design and implement a database system
    - Storage management
    - Query processing
    - Transaction management

The Big Picture-- DBMS Architecture



User/Web Forms/Applications/DBA

query

transaction

DDL commands

Query Parser

Query Rewriter

Query Optimizer

Query Executor

Records

Indexes

Buffer Manager

Storage Manager

Transaction Manager

Concurrency Control

DDL Processor

Logging & Recovery

Buffer: data, indexes, log, etc

Lock Tables

Main Memory

Storage

data, metadata, indexes, log, etc

Disks
Buffer Manager

The Memory Hierarchy (2008)

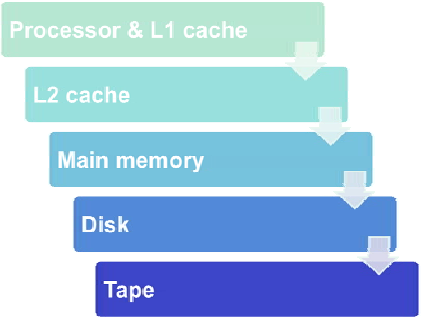
Processor Cache:
- access time = 1-3 nanosecs.

Main Memory = Disk Cache
- Volatile
- a few GB
- expensive
- Access time: 10-100 nanosecs

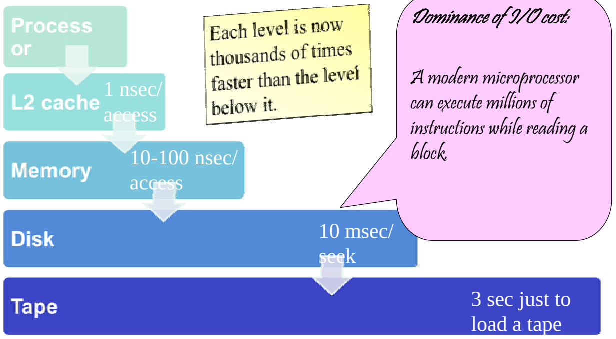
Disk
- Persistent
- 1 TB storage
- speed:
  - Rate=5-10 MB/S
  - Access time = 10 msecs.

Tape
- 1.5 MB/S transfer rate
- Only sequential access
- Not for operational data

The memory hierarchy



The relative gaps in performance are increasing.



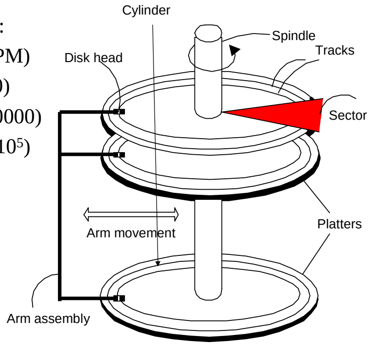
Each level is now thousands of times faster than the level below it.

Dominance of I/O cost:

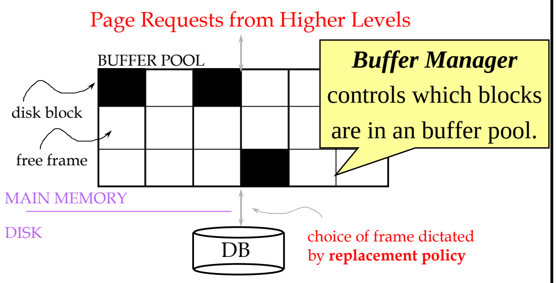
A modern microprocessor can execute millions of instructions while reading a block.

The Mechanics of Disk

Mechanical characteristics:
- Rotation speed (5400RPM)
- Number of platers (1-30)
- Number of tracks (<=10000)
- Number of bytes/track(10^5)



Buffer Management in a DBMS



Buffer Manager controls which blocks are in an buffer pool.

- Files are moved between disk and main memory in blocks; it takes roughly 10 milliseconds
- It is vital that a disk block we are accessing is already in a buffer pool!

Representing Data

Terminology in Secondary Storage

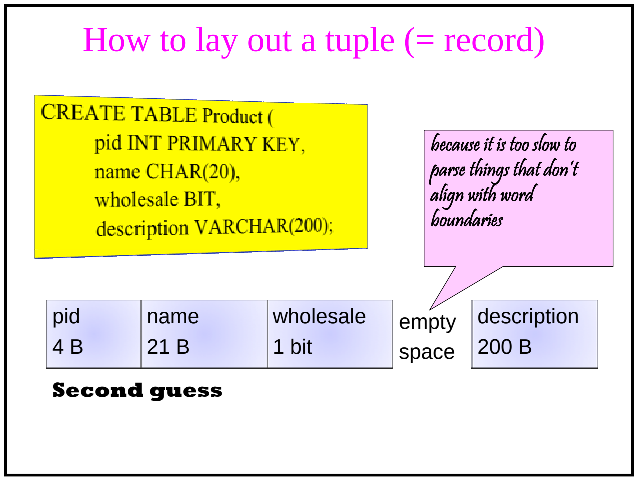
	Data element	Record	Collection
SQL	attribute	tuple	relation
Files	field	record	file

How to lay out a tuple (= record)

```
CREATE TABLE Product (
    pid INT PRIMARY KEY,
    name CHAR(20),
    wholesale BIT,
    description VARCHAR(200);
```

pid 4 B	name 21 B	wholesale 1 bit	description 200 B

First guess



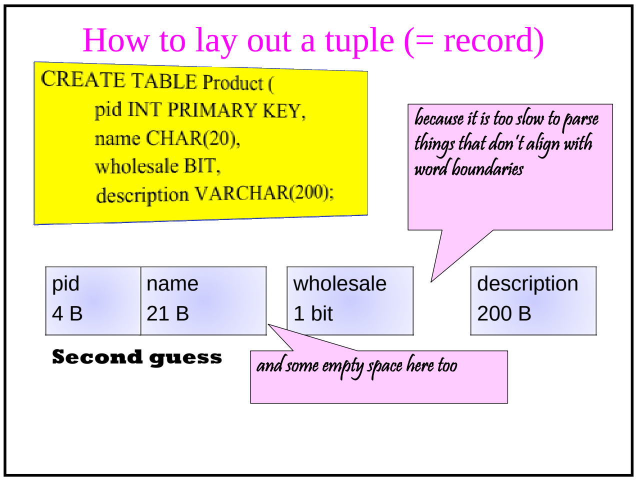
How to lay out a tuple (= record)

```
CREATE TABLE Product (
        pid INT PRIMARY KEY,
        name CHAR(20),
        wholesale BIT,
        description VARCHAR(200);
```

because it is too slow to parse things that don't align with word boundaries

pid	name	wholesale	empty	description
4 B	21 B	1 bit	space	200 B

Second guess



How to lay out a tuple (= record)

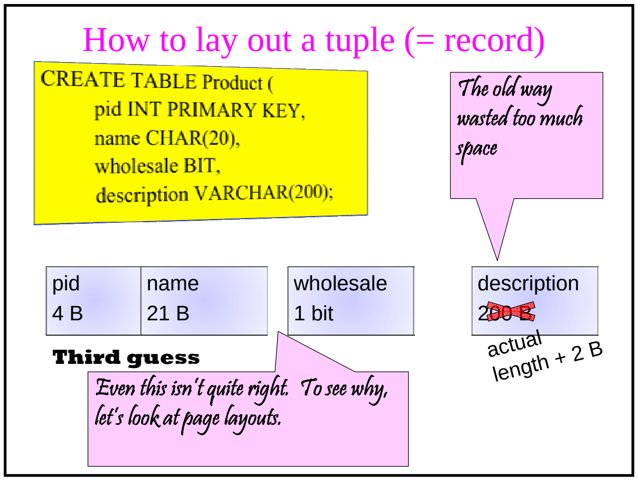
```
CREATE TABLE Product (
        pid INT PRIMARY KEY,
        name CHAR(20),
        wholesale BIT,
        description VARCHAR(200);
```

because it is too slow to parse things that don't align with word boundaries

pid	name	wholesale	description
4 B	21 B	1 bit	200 B

Second guess

and some empty space here too



How to lay out a tuple (= record)

```
CREATE TABLE Product (
        pid INT PRIMARY KEY,
        name CHAR(20),
        wholesale BIT,
        description VARCHAR(200);
```

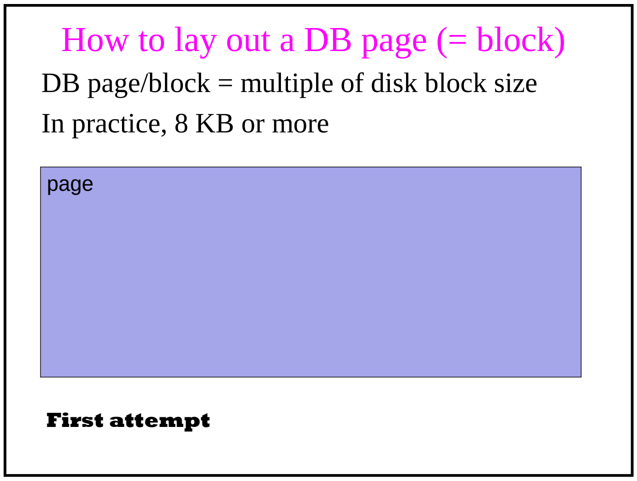
The old way wasted too much space

pid	name	wholesale	description
4 B	21 B	1 bit	~~200 B~~

actual length + 2 B

Third guess

Even this isn't quite right. To see why, let's look at page layouts.



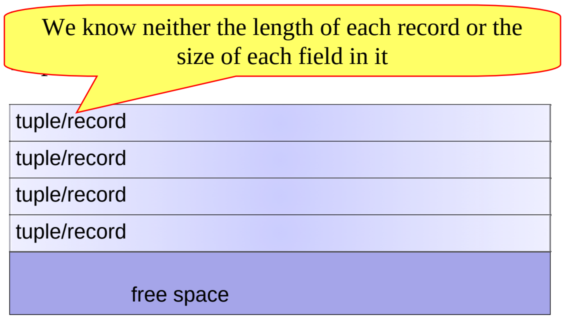
How to lay out a DB page (= block)

DB page/block = multiple of disk block size

In practice, 8 KB or more

First attempt

How to lay out fixed-length records

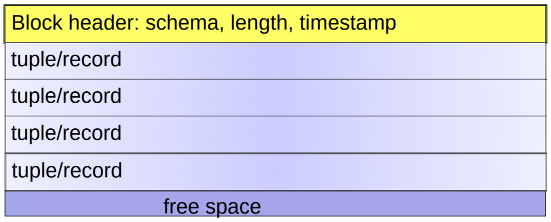

We know neither the length of each record or the size of each field in it

First attempt

How to lay out fixed-length records

DB page/block = multiple of disk block size

In practice, 8 KB or more

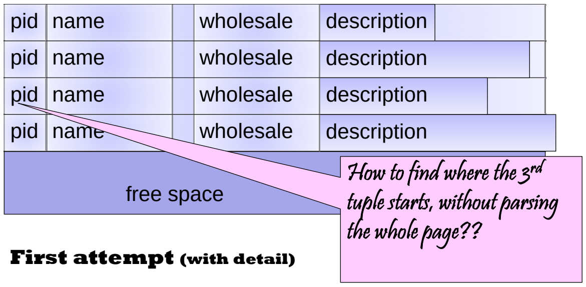


Second attempt

How to lay out variable-length records

DB page/block = multiple of disk block size

In practice, 8 KB or more


pid name wholesale description
pid name wholesale description
pid name wholesale description
pid name wholesale description
free space

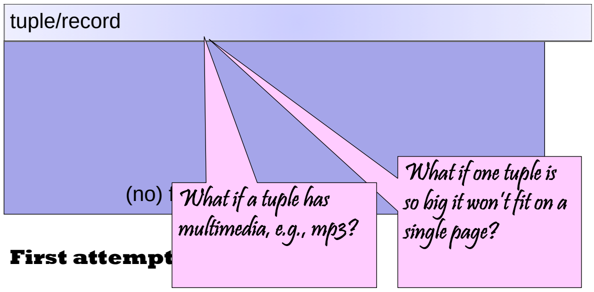
How to find where the 3rd tuple starts, without parsing the whole page??

First attempt (with detail)

How to handle huge records?

DB page/block = multiple of disk block size = 8 KB+

Need a tuple? Fetch its ***entire*** page into memory.

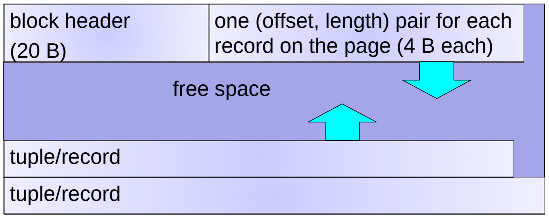


What if a tuple has multimedia, e.g., mp3?

What if one tuple is so big it won't fit on a single page?

First attempt

How to lay out variable-length records



block header (20 B)

one (offset, length) pair for each record on the page (4 B each)

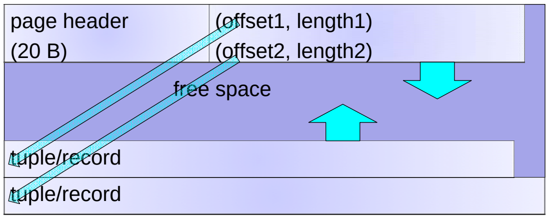
free space

tuple/record

tuple/record

Refer to a tuple as (page#, i) ***for its entire lifetime***, even though the DBMS rearranges page contents

How to lay out variable-length records



page header (20 B)

(offset1, length1)

(offset2, length2)

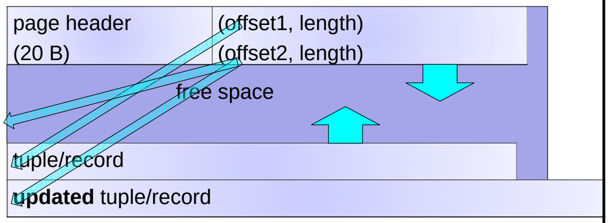
free space

tuple/record

tuple/record

Refer to a tuple as (page#, offset id) ***for its entire lifetime***, even though the DBMS rearranges page contents

Why rearrange a DB page?



page header (20 B)

(offset1, length)

(offset2, length)

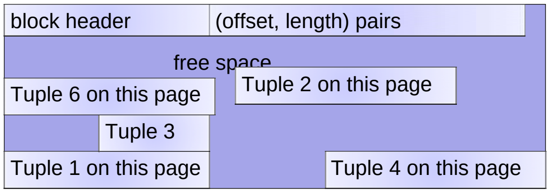
free space

tuple/record

updated tuple/record

In most DBMSs, all the tuples on a page will be from the same relation.

Eventually the free space may be so fragmented that you'll need to defragment



block header

(offset, length) pairs

free space

Tuple 6 on this page

Tuple 2 on this page

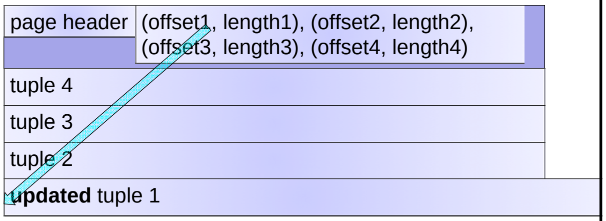
Tuple 3

Tuple 1 on this page

Tuple 4 on this page

In practice, that doesn't happen very often, because most applications tend to get more and more data.

What if a tuple no longer fits on the page?



page header | (offset1, length1), (offset2, length2), (offset3, length3), (offset4, length4)

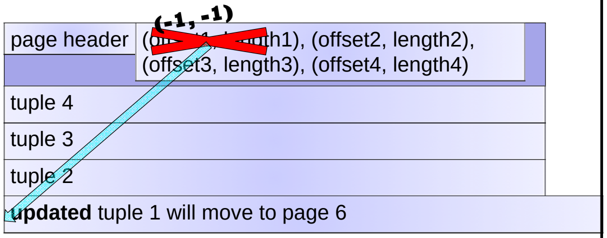
tuple 4

tuple 3

tuple 2

updated tuple 1

What if a tuple no longer fits on the page?



page header | (~~offset1, length1~~) (-1, -1), (offset2, length2), (offset3, length3), (offset4, length4)

tuple 4

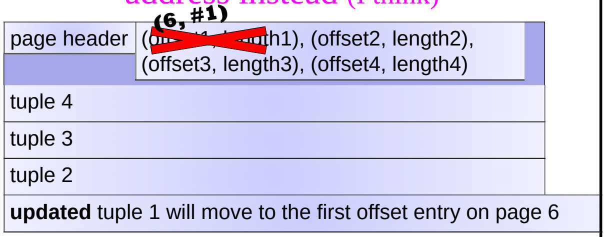
tuple 3

tuple 2

updated tuple 1 will move to page 6

If you just move it to a new page, you must find & fix the dangling “pointers” to it in indexes & memory.

Some DBMSs leave a forwarding address instead (I think)



page header | (~~offset1, length1~~) (6, #1), (offset2, length2), (offset3, length3), (offset4, length4)

tuple 4

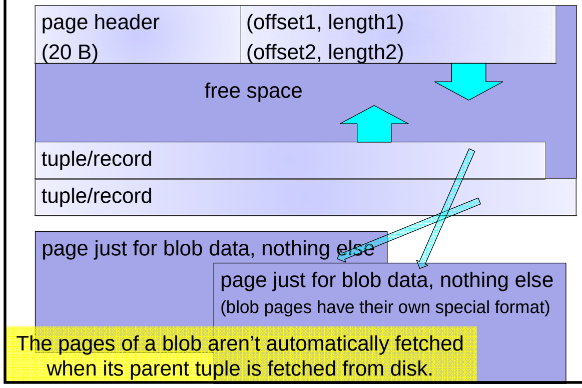
tuple 3

tuple 2

updated tuple 1 will move to the first offset entry on page 6

Don’t need to find/fix dangling pointers, but every access to the relocated tuple will take twice as long

Where do Binary Large Objects (BLOBs) go? (mp3s, jpegs, …)



page header (20 B) | (offset1, length1) (offset2, length2)

free space

tuple/record

tuple/record

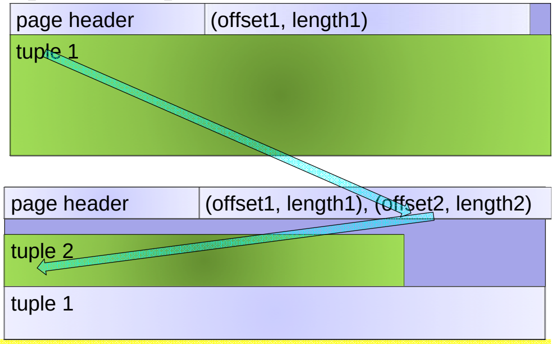
page just for blob data, nothing else

page just for blob data, nothing else (blob pages have their own special format)

The pages of a blob aren’t automatically fetched when its parent tuple is fetched from disk.

What about tuples bigger than a page?

***spanned* tuples**



page header

(offset1, length1)

tuple 1

page header

(offset1, length1), (offset2, length2)

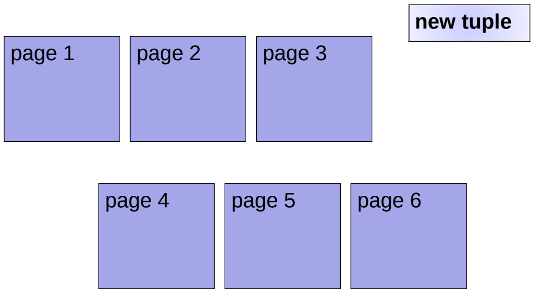
tuple 2

tuple 1

You should ***seriously*** consider changing the DB page size.

Record Modifications

Insertions are easy if the file isn't stored sorted on some field (e.g., primary key)



new tuple

page 1

page 2

page 3

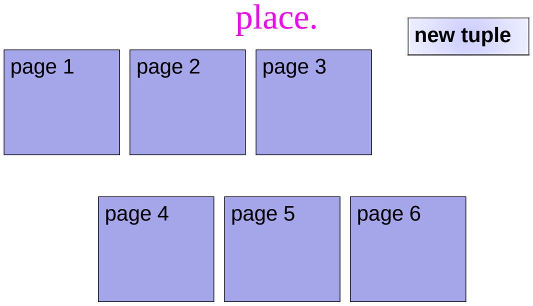
page 4

page 5

page 6

Put the new tuple at the end of the file.

If the file is stored sorted on some field, then the DBMS has to put it in the right place.



new tuple

page 1

page 2

page 3

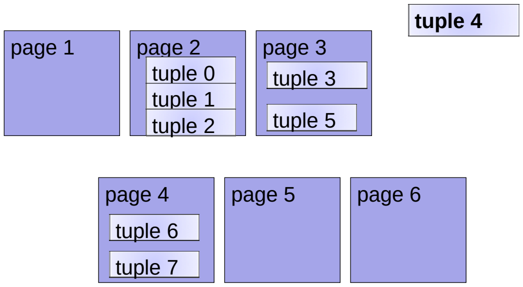
page 4

page 5

page 6

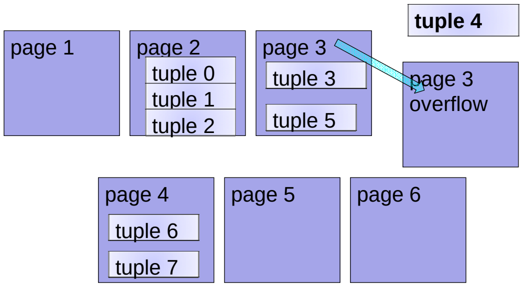
But what if there is no room on that page?

The DBMS can try to rearrange nearby pages to make room.



But those pages may be filled also.

An alternative is to create an ***overflow page*** for the too-full page**.**



To keep good performance, the DBMS must occasionally rebuild the entire file to merge in the overflow pages.

In reality, deletions are rare in DB apps.

But if you have a deletion:

- Free up space in its block
- Possibly eliminate an overflow block
- Can’t shrink the (offset, length) array, but may be able to recycle the old tuple’s slot for a new tuple

What if indexes/logs/other things may still point to the deleted record?

- Place a ***tombstone*** instead (a NULL record, or a special (offset, length) entry)