TODAY

• Lecture on chapter 3 for ~30 min

• Then introduction to project 1 ~20 min

• Project-1 team formation later today (watch for it on my.wpi)
Storage Layout:
How to lay out data on disk.
( chapter 3)
Overview

Data Items
  ↓
  Records
  ↓
  Blocks
  ↓
  Files
  ↓
  Memory
Record - Collection of related data items (called FIELDS)

E.g.: Employee record:
  name field,
  salary field,
  date-of-hire field,
  ...

Types of records:

- Main choices:
  - FIXED vs VARIABLE FORMAT
  - FIXED vs VARIABLE LENGTH
A SCHEMA contains information such as:
- # fields (attributes)
- type of each field (length)
- order of attributes in record
- meaning of each field (domain)
- constraints (primary key, etc).

Not associated with each record.
Example: fixed format and length

Employee record

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

We can simply concatenate fields.
Variable format

• **What**: Not all fields are included in the record, and possibly in different orders.

• **Then**: Record itself must contain format, i.e., it is “Self Describing”: 
Why Variable format?

- “sparse” records
- repeating fields
- evolving formats
Example: variable format and length

Field name codes could also be strings, i.e. TAGS
• **EXAMPLE:**

variable format record with repeating fields

e.g., Employee has one or more children

| 3 | E_name: Fred | Child: Sally | Child: Tom |

• Do repeating fields always require variable format and length?
• Then must allocate maximum number of repeating fields (if not used, set to null)

e.g., a person and her hobbies.

<table>
<thead>
<tr>
<th>Mary</th>
<th>Sailing</th>
<th>Chess</th>
<th>--</th>
</tr>
</thead>
</table>
Many variants between fixed - variable format:

Example 1: Include record type in record

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>27</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

record type tells me what
to expect
(i.e., points to schema)
Record header - data at beginning that describes record

May contain:
- pointer to schema (record type)
- length of record
- time stamp (create time, mod. time)
- other stuff (e.g., ROW-ID in Oracle)
Example 2: Variant btw FIXED/VAR format

- Hybrid format: one part is fixed, other is variable

E.g.: All employees have E#, name, dept; and other fields vary.

| 25 | Smith | Toy | 2 | Hobby : chess | retired |

# of var fields
Also, many variations in internal organization of record

Just to show one:

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>F2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>F3</td>
<td>12</td>
<td>5</td>
</tr>
</tbody>
</table>

total size: 32

offsets: 0, 1, 2, 3, 4, 5, 15, 20

length of field: *
Question:
We have seen examples for:
* Fixed format and length records
* Variable format and length records

(a) Does fixed format and variable length make sense?
(b) Does variable format and fixed length make sense?
Next:

Data Items

↓

Records

↓

Blocks

↓

Files

↓

Memory
Next: placing records into blocks

blocks ...

assume fixed length blocks

a file ← assume a single file (for now)
Options for storing records in blocks:

(1) separating records
(2) spanned vs. unspanned
(3) mixed record types – clustering
(4) split records
(5) sequencing
(6) 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

  ![Unspanned Diagram]

- **Spanned**

  ![Spanned Diagram]
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)
brack size = 4096 bytes

<table>
<thead>
<tr>
<th>block 1</th>
<th>block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>R2</td>
</tr>
<tr>
<td>2050 bytes wasted 2046</td>
<td>2050 bytes wasted 2046</td>
</tr>
</tbody>
</table>

• Utiliz = 50% -> ½ of space is wasted
(3) Mixed record types

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

  e.g., a block

<table>
<thead>
<tr>
<th>EMP</th>
<th>e1</th>
<th>DEPT</th>
<th>d1</th>
<th>DEPT</th>
<th>d2</th>
</tr>
</thead>
</table>
Why do we want to mix?

• Answer: CLUSTERING

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

• Problems

  Creates variable length records in block
  Must avoid duplicates (how to cluster?)
  Insert/deletes are harder
Example Clustering

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

a block

CUSTOMER,NAME=SMITH
DEPOSIT,NAME=SMITH
DEPOSIT,NAME=SMITH
• If Q1 frequent, clustering good
• But if Q2 frequent
  Q2: SELECT * FROM CUSTOMER

CLUSTERING IS COUNTER PRODUCTIVE
Compromise:

No mixing, but keep related records in same cylinder ...
Recap: Storing records in blocks

(1) Separating records
(2) Spanned vs. Unspanned
(3) Mixed record types - Clustering
(4) Split records
(5) Sequencing
(6) Indirection
Now on to Project 1 ...