Storing variable length data/records:

(1) Data item has variable size
   e.g.: string

(2) Repeating field:
   e.g.: representing a many-many relationship inside a record.

Employee | SSN | Addr | Pn1, Pn2, Pn3, ...

Repeating field

(3) Variable format records
   e.g.: XML:
   
   <Name>
   John Smith
   </Name>

(4) Large size fields
   "Binary Large Objects" (BLOB)
   e.g.: music, .swf
Storing variable sized records in a block:

- Recall: Fixed sized records:
  - Header
  - Links to other blocks
  - Info about role of data in this block
  - Offset of records in block

Block with variable length records:

- Header
- Null unused offsets
- Unused records
- Unused records
- Unused records
- Offset table
- Record space
- Growth direction
Storing a variable sized record itself

Person:

- Name
- Address
- Gender
- Variable length
- Fixed length
- Bdate

Record stored as:

1. Put fixed sized field first.
2. First variable sized field start right after all the fixed sized fields.
3. Offset pointer separate field
4. Record length ends last var. sized field

Record length
Another example:

**Person:**

```
| Name | G | Address | Bdate |
```

- Fixed size
- Variable size

Stored as:
- Put fixed sized field first

Record length
- First field
- Start after all fixed sized field
Storing records with repeating fields

Example:

Movie Star:

<table>
<thead>
<tr>
<th>gender (fixed size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
</tbody>
</table>

Variable size

repeating field (fixed size, each one)

Stored as:

Record header

<table>
<thead>
<tr>
<th>Fixed length first</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

record length

First

variable length field

starts

(i3 known!)

implicit
Note: If you have MORE variable/ repeating field, the END can always be detected by the MDX in the record header.

- The MDX/pointer marks the end of a variable/ repeating field.
Alternative Storage Structure for Variable Length and Repeating Fixed Records:

1. Use 2 separate records.
   a. A fixed length record with data and pointers to variable length fields.
   b. The variable length field(s) are stored separately.

E.g.:

- **Record header:**
  - Name
  - Length of Name
  - Address
  - Movies

- **Diagram:**
  - Connections between fields
  - Variables and pointers illustrated
Advantages:

- The record is fixed length

  \( \rightarrow \) allow record to be moved more easily.

Disadvantage:

- More disk access to access variable length field

  (Need 2 disk accesses)
Hybrid approach:

- Waste some space
- Faster access for small size # of repeats.

repetitive fixed
≤ N repeats.

fixed size

variable size

≤ MAX length.
Storing variable for MAT records

- When you store a record whose structure is unknown/variable

you must:

1. Define an ontology (Naming scheme)

   \( \text{eg: } N = \text{Name} \)
   \( \text{R = Restaurant owned} \)
   \( \text{S = String} \)
   \( \text{I = Integer} \)

2. Store the meaning of a field (encoded with the ontology) along with the value of the field.
**Typical entry format of an entry for one field**

**Meta Information**

<table>
<thead>
<tr>
<th>Meaning of the field</th>
<th>Data Type used to store values</th>
<th>Length (in # of bytes)</th>
<th>Value of the field</th>
</tr>
</thead>
</table>

**Eg:**

- **Meta Info:**
  - It's strong.
- **This is a Name:**
- **This is a Restaurant owned by this person:**

Assume every field is 1 byte in length.
Records that are larger than a block

- Spanning record = a record that is stored inside > 1 block.

NB: records that are smaller than a block CAN span > 1 block.

Eg: To store records efficiently.

\[ \text{1 block} = 64 \text{K} \]

\[ \text{1 record} = 32 \text{K} \]

\[ \Rightarrow \text{Waste too much space if we store} \]

\[ 1 \text{ record in one block} !!! \]
Storing a record that is spread over 2 blocks:

Record Header stores:

1. A bit to indicate whether the record is whole or fragmented.

2. (2 pieces): A bit to tell if fragment is:
   - first fragment
   - or last fragment

   (Use a fragment # if more)

3. Pointers to previous and next fragments. (In each fragment)
Storing BLOBs:

Part in the same cylinder to reduce seek time.

Retrieves only portion of the BLOB

(E.g., movie: enough to play at the display rate)