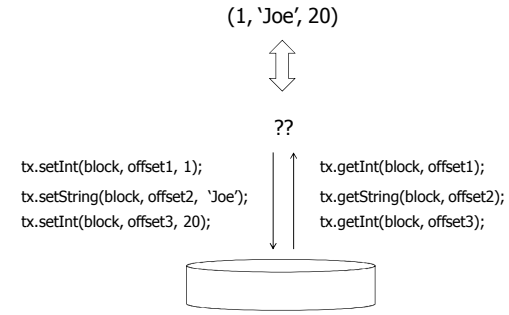
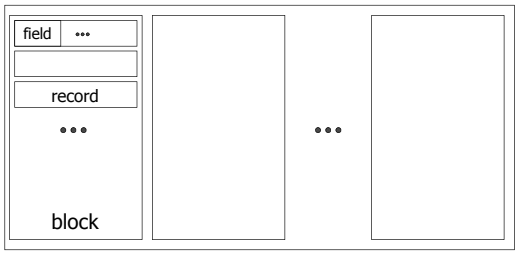
CS422 Principles of Database Systems
Record Management

Chengyu Sun
California State University, Los Angeles

Record Management



Records on Disk

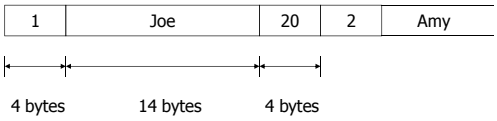


Simple Assumptions

- Each table is stored in its own file
- All fields are fixed-length
- Each record is contained in one block

Records in a File

- Record <int, varchar(10), int>
  - <1,'Joe',20> and <2, 'Amy', 10>
- *Would this work??*

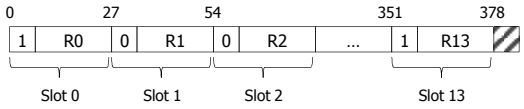


Supported Operations

- Retrieve
- Insert
- Update
- Delete

Record Block

- A block is formatted into *slots*
- Slot = Record + 1 byte (empty flag)
- Example
  - Record length 26 bytes
  - Block size 400 bytes



Record Management in SimpleDB

- `simpledb.record`

Table Information

- `Schema`
  - A collection of *fields*
  - Each field has name, type, and length
- `TableInfo`
  - Schema
  - Physical information of the table
    - File name
    - Record length
    - Field offsets

Access Records in a Block

- `RecordFormatter`
  - Format an empty page, i.e. create the *slots*
- `RecordPage`
  - Pin a block
  - Access a record
    - Move to a slot
    - Get/set each field
    - Delete

Access Records in a Table

- `RecordFile`
  - Keep one block in memory in a RecordPage
  - next() will load in the next block if the current block has no more slots to return
  - `RID` = block number + slot number

Example: Access Records

- Use SQLInterpreter
  - Create a table students(sid, sname, dept)
  - Insert two records
- Use a program to
  - Read the records in the table
  - Delete, update, insert

Extend the Basic Scheme

- Variable-length fields
- Spanned records
- Non-homogeneous files

Store Variable-Length Fields

(a) 1 Joe 20 | 2 Samantha 20 | 3 John 10

(b) 1 0 20 | 2 3 20 | 3 11 10

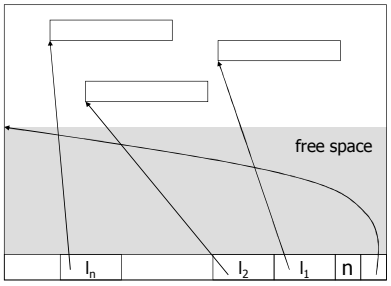
JoeSamanthaJohn

0 3 11

(c) 1 Joe 20 | 2 Samantha 20 | 3 John 10

Appropriate approach for char(n), varchar(n), clob??

Block Organization



Why ID-table is stored at the end of a block??

Support Operations for Variable-length Fields

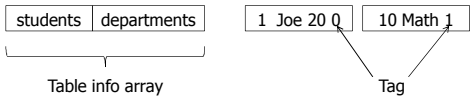
- Retrieve
- Insert
- Delete
- Update
  - Overflow block

Spanned Records

- Records that span multiple blocks
- Record header
  - Indicates whether the record is a fragment
  - Pointers to the next/previous fragment

Non-homogeneous Files

- A file contains records from different tables
  - Records within a block are from the same table
  - Records within a block are from different tables
- Implement non-homogeneous files
  - Table-block directory
  - Table info array and record tag, e.g.



Readings

- Textbook Chapter 15
- SimpleDB source code
  - `simpledb.record`