A

CSE 444: Database Internals.
Section 2: Data Storage and SQL.

1. SQL

In this part, we'll briefly review SQL on the whiteboard (no computer). The database is as follows:

(1) Crimes(offence_id, location, type, date)
(2) Checkins(name, num_checkins)
(3) Locations(name, location, zipcode)

1.1. Review of joins

(1) List the name and the number of checkins for all businesses in zipcode 98105.
(2) List the offense_id and type of all crimes reported in zipcode 98105 on 12/31/2011.

1.2. Queries using aggregation and grouping

(1) List the count of burglary reported in zipcode 98105.
(2) List all the different crimes and their count in zipcode 98105.

1.3. Queries with joins, grouping, and aggregation

List the name of the business, its number of checkins, and the count of all crimes that have occurred at all businesses in zipcode 98105.

[One possible solution for the above query is:]

```
SELECT chk.name, chk.num_checkins, count(cr.offence_id) as n_crimes
FROM Checkins chk, Crimes cr, Locations l
WHERE chk.zipcode = 98105 AND
      chk.name = l.name AND
      cr.location = l.location
GROUP BY chk.name, chk.num_checkins
```

1.4. Query Tree

Consider the simplest tree for the query with joins, grouping, and aggregation: all joins, followed by grouping, followed by all projections, followed by aggregation, and then followed by all selections.

What if we push down selection?
What if we change the join order?
What if we push down projections?

2. VARIABLE LENGTH RECORDS

13.7.1, Molina et al.

A patient record consists of the following fixed-length fields: the patient's date of birth, social-security number, and patient ID, each 10 bytes long. It also has the following variable-length fields: name, address, and patient history. If pointers within a record require 4 bytes, and the record length is a 4-byte integer, how many bytes, exclusive of the space needed for the variable-length fields, are needed for the record? You may assume no alignment of fields is required.

[4 (for the record length) + 4 * 2 (for the pointers to the beginning of the second and the third variable-length field) + 10 * 3 (for the fixed-length fields) = 42 bytes]

9.18, Ramakrishnan et al.

Consider the page format for variable-length records that uses a slot directory.

(1) One approach to managing the slot directory is to use a maximum size (*i.e.*, a maximum number of slots) and allocate the directory array when the page is created. Discuss the pros and cons of this approach with respect to the approach discussed in the text. [Cons: This, very likely, wastes space.] [Pros: For insert only workload, this could be faster. This is because there is no need to increment the size of the slot directory after each insert. Another advantage? Coming up!]
(2) Suggest a modification to this page format that would allow us to sort records (according to the value in some field) without moving records and without changing the record ids. [Create another directory of slots, point to the slot used for the record id, add entries in the sorted order of the value of the field we are interested in.]

9.19, Ramakrishnan et al.

Consider the two internal organizations for heap files (using lists of pages and a directory of pages) discussed in the text.

(1) What are the trade-offs? Which organization would you choose if records are variable in length? [Directory of pages is preferred since we can store the free space left in each page in the directory itself.]
(2) Can you suggest a single page format to implement both internal file organization? [Allocate a bit to indicate the page type, and two pointers per page (for creating the linked list).]