CS186 Week 1: SQL, Buffer Pool Replacement Policies

Lu Cheng
lu.cheng@berkeley.edu
Office Hours: Thursday 4-5 PM
Location: TBD

What is a database?

What is a database?

a structured set of data held in a computer, esp. one that is accessible in various ways.

Basic SQL

```
CREATE TABLE
Students
(sid CHAR(20),
 name CHAR(20),
 login CHAR(10),
 age INTEGER,
 gpa FLOAT)
```

Basic SQL

```
CREATE TABLE
Students
(sid CHAR(20),
 name CHAR(20),
 login CHAR(10),
 age INTEGER,
 gpa FLOAT)
```

SID	Name	Login	Age	GPA

Data types in SQL

- INT (32 bits)
- SMALLINT (16 bits)
- TINYINT (8 bits)
- BIGINT (64 bits)
- FLOAT
- DOUBLE
- DATE
- DATETIME
- TIMESTAMP
- CHAR
- VARCHAR

Basic SQL

INSERT INTO Students (sid, name, login, age, gpa)
VALUES ('53688', 'Smith', 'smith@ee', 18, 1.2)

SID	Name	Login	Age	GPA

Basic SQL

INSERT INTO Students (sid, name, login, age, gpa)
VALUES ('53688', 'Smith', 'smith@ee', 18, 1.2)

SID	Name	Login	Age	GPA
‘53688’	Smith	smit@ee	18	1.2

Basic SQL

INSERT INTO Students (sid, name, login, age, gpa)
VALUES ('53689', ‘Joe', ‘joe@cs', 18, 4.5)

SID	Name	Login	Age	GPA
‘53688’	Smith	smit@ee	18	1.2
‘53689’	Joe	joe@cs	18	4.5

Basic SQL

And so on...

SID	Name	Login	Age	GPA
‘53688’	Smith	smit@ee	18	1.2
‘53689’	Joe	joe@cs	18	4.5
‘123’	John	john@ee	19	3.5
‘567’	Jill	Jill@cs	20	3.6
‘891’	Jack	Jack@ee	25	3.7

Basic SQL

```
SELECT *
    FROM Students S
WHERE S.age=18
    AND S.gpa > 2.0
```

SID	Name	Login	Age	GPA
‘53688’	Smith	smit@ee	18	1.2
‘53689’	Joe	joe@cs	18	4.5
‘123’	John	john@ee	19	3.5
‘567’	Jill	Jill@cs	20	3.6
‘891’	Jack	Jack@ee	25	3.7

Basic SQL

```
SELECT *
    FROM Students S
WHERE S.age=18
    AND S.gpa > 2.0
```

SID	Name	Login	Age	GPA
‘53688’	Smith	smit@ee	18	1.2
‘53689’	Joe	joe@cs	18	4.5
‘123’	John	john@ee	19	3.5
‘567’	Jill	Jill@cs	20	3.6
‘891’	Jack	Jack@ee	25	3.7

Basic SQL

```
SELECT name, sid
    FROM Students S
WHERE S.age=18
    AND S.gpa > 2.0
```

SID	Name	Login	Age	GPA
‘53688’	Smith	smit@ee	18	1.2
‘53689’	Joe	joe@cs	18	4.5
‘123’	John	john@ee	19	3.5
‘567’	Jill	Jill@cs	20	3.6
‘891’	Jack	Jack@ee	25	3.7

Joining Tables

Given these two tables below, join them to create tuples of the form (sid, name, course_name, grade)

```
CREATE TABLE Students
(sid CHAR(20),
 name CHAR(20),
 gpa FLOAT)
```

```
CREATE TABLE
Grades
(sid CHAR(20),
 course_name CHAR(20),
 grade CHAR(2)
 Foreign Key (sid)
REFERENCES Students
(sid))
```

Joining Tables

Given these two tables below, join them to create tuples of the form (sid, name, course_name, grade)

Students

SID	Name	GPA
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	Course Name	Grade
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘0’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘Jack’	‘A’

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	Name	GPA
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	Course Name	Grade
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘0’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘Jack’	‘A’

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	**Name**	**GPA**
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	**Course Name**	**Grade**
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘0’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘CS70’	‘A’

Joined

SID	Name	Course Name	Grade

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	Name	GPA
'0'	Smith	1.2
'1'	Joe	4.5
'2'	John	3.5
'3'	Jill	3.6
'4'	Jack	3.7

Grades

SID	Course Name	Grade
'0'	'CS61A'	'A'
'2'	'CS186'	'A+'
'0'	'CS186'	'B-'
'0'	'CS188'	'C-'
'1'	'CS70'	'A'

Joined

SID	Name	Course Name	Grade
'0'	'Smith'	'CS61A'	'A'

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	Name	GPA
'0'	Smith	1.2
'1'	Joe	4.5
'2'	John	3.5
'3'	Jill	3.6
'4'	Jack	3.7

Grades

SID	Course Name	Grade
'0'	'CS61A'	'A'
'2'	'CS186'	'A+'
'0'	'CS186'	'B-'
'0'	'CS188'	'C-'
'1'	'CS70'	'A'

Joined

SID	Name	Course Name	Grade
'0'	'Smith'	'CS61A'	'A'

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	Name	GPA
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	Course Name	Grade
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘0’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘CS70’	‘A’

Joined

SID	Name	Course Name	Grade
‘0’	‘Smith’	‘CS61A’	‘A’
‘0’	‘Smith’	‘CS186’	‘B-’

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	Name	GPA
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	Course Name	Grade
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘0’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘CS70’	‘A’

Joined

SID	Name	Course Name	Grade
‘0’	‘Smith’	‘CS61A’	‘A’
‘0’	‘Smith’	‘CS186’	‘B-’
‘0’	‘Smith’	‘CS188’	‘C-’

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	Name	GPA
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	Course Name	Grade
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘0’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘CS70’	‘A’

Joined

SID	Name	Course Name	Grade
‘0’	‘Smith’	‘CS61A’	‘A’
‘0’	‘Smith’	‘CS186’	‘B-’
‘0’	‘Smith’	‘CS188’	‘C-’

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	Name	GPA
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	Course Name	Grade
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘0’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘CS70’	‘A’

Joined

SID	Name	Course Name	Grade
‘0’	‘Smith’	‘CS61A’	‘A’
‘0’	‘Smith’	‘CS186’	‘B-’
‘0’	‘Smith’	‘CS188’	‘C-’
‘1’	‘Joe’	‘CS70’	‘A’

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	Name	GPA
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	Course Name	Grade
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘0’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘CS70’	‘A’

Joined

SID	Name	Course Name	Grade
‘0’	‘Smith’	‘CS61A’	‘A’
‘0’	‘Smith’	‘CS186’	‘B-’
‘0’	‘Smith’	‘CS188’	‘C-’
‘1’	‘Joe’	‘CS70’	‘A’
‘2’	‘John’	‘CS186’	‘A+’

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	Name	GPA
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	Course Name	Grade
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘0’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘CS70’	‘A’

Joined

SID	Name	Course Name	Grade
‘0’	‘Smith’	‘CS61A’	‘A’
‘0’	‘Smith’	‘CS186’	‘B-’
‘0’	‘Smith’	‘CS188’	‘C-’
‘1’	‘Joe’	‘CS70’	‘A’
‘2’	‘John’	‘CS186’	‘A+’

Joining Tables

SELECT sid, name, course_name, grade FROM Students S, Grades G WHERE S.sid = G.sid;

Students

SID	Name	GPA
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	Course Name	Grade
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘0’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘CS70’	‘A’

Joined

SID	Name	Course Name	Grade
‘0’	‘Smith’	‘CS61A’	‘A’
‘0’	‘Smith’	‘CS186’	‘B-’
‘0’	‘Smith’	‘CS188’	‘C-’
‘1’	‘Joe’	‘CS70’	‘A’
‘2’	‘John’	‘CS186’	‘A+’

Joining Tables

What if we had a tuple in Grades with SID = ‘5’

Students

SID	Name	GPA
‘0’	Smith	1.2
‘1’	Joe	4.5
‘2’	John	3.5
‘3’	Jill	3.6
‘4’	Jack	3.7

Grades

SID	Course Name	Grade
‘0’	‘CS61A’	‘A’
‘2’	‘CS186’	‘A+’
‘5’’	‘CS186’	‘B-’
‘0’	‘CS188’	‘C-’
‘1’	‘CS70’	‘A’

Joined

SID	Name	Course Name	Grade
‘0’	‘Smith’	‘CS61A’	‘A’
‘0’	‘Smith’	‘CS186’	‘B-’
‘0’	‘Smith’	‘CS188’	‘C-’
‘1’	‘Joe’	‘CS70’	‘A’
‘2’	‘John’	‘CS186’	‘A+’

Buffer Pools

- Set of pages reserved for memory management
  - Can't fit all of database in memory
  - Fit some relevant subset
- What's "relevant"?
  - Stuff we need to operate on right now (working set)
  - Stuff we may need to operate on in the near future (cache)

Page

- Don’t want to read database from memory all at once and database usually doesn’t fit into memory.
- If you want to read a single tuple (sid, name, gpa), you’d have to load the entire page that the tuple is in. (Doesn’t make much sense)
- However, you’re likely going to read other things from that page. (Spatial Locality)
- Page size is important

Filling the Buffer Pool

Consider a table Student that is stored across 10 pages with 50 tuples each. Let's say you have 5 buffers (A, B, C, D, E).

What happens to the buffer pools when you run the query: SELECT * FROM student?

	1	2	3	4	5	6	7	8	9	10
A	1									
B										
C										
D										
E										

Filling the Buffer Pool

Consider a table Student that is stored across 10 pages with 50 tuples each. Let's say you have 5 buffers (A, B, C, D, E).

What happens to the buffer pools when you run the query: SELECT * FROM student?

	1	2	3	4	5	6	7	8	9	10
A	1									
B		2								
C										
D										
E										

Filling the Buffer Pool

Consider a table Student that is stored across 10 pages with 50 tuples each. Let's say you have 5 buffers (A, B, C, D, E).

What happens to the buffer pools when you run the query: SELECT * FROM student?

	1	2	3	4	5	6	7	8	9	10
A	1									
B		2								
C			3							
D										
E										

Filling the Buffer Pool

Consider a table Student that is stored across 10 pages with 50 tuples each. Let's say you have 5 buffers (A, B, C, D, E).

What happens to the buffer pools when you run the query: SELECT * FROM student?

	1	2	3	4	5	6	7	8	9	10
A	1									
B		2								
C			3							
D				4						
E										

Filling the Buffer Pool

Consider a table Student that is stored across 10 pages with 50 tuples each. Let's say you have 5 buffers (A, B, C, D, E).

What happens to the buffer pools when you run the query: SELECT * FROM student?

	1	2	3	4	5	6	7	8	9	10
A	1									
B		2								
C			3							
D				4						
E					5					

Filling the Buffer Pool

UH OH. What now?

Let's replace pages using a LRU policy

We remove pages from the buffer pool that were least recently accessed/accessed the longest time ago. (Temporal locality)

	1	2	3	4	5	6	7	8	9	10
A	1									
B		2								
C			3							
D				4						
E					5					

Filling the Buffer Pool

Consider a table Student that is stored across 10 pages with 50 tuples each. Let's say you have 5 buffers (A, B, C, D, E).

What happens to the buffer pools when you run the query: SELECT * FROM student?

	1	2	3	4	5	6	7	8	9	10
A	1					6				
B		2								
C			3							
D				4						
E					5					

Filling the Buffer Pool

Consider a table Student that is stored across 10 pages with 50 tuples each. Let's say you have 5 buffers (A, B, C, D, E).

What happens to the buffer pools when you run the query: SELECT * FROM student?

	1	2	3	4	5	6	7	8	9	10
A	1					6				
B		2					7			
C			3							
D				4						
E					5					

Filling the Buffer Pool

Consider a table Student that is stored across 10 pages with 50 tuples each. Let's say you have 5 buffers (A, B, C, D, E).

What happens to the buffer pools when you run the query: SELECT * FROM student?

	1	2	3	4	5	6	7	8	9	10
A	1					6				
B		2					7			
C			3					8		
D				4					9	
E					5					10

Read 1 MB in 1 KB Blocks

- Disk seek 10,000,000 ns (10 milliseconds)
- Read 1 MB sequentially from disk (30 milliseconds)
- How long does it take to read 1 MB if the data is only sequential in 1 KB blocks? (Have to do a disk seek for every KB)

Sequential Flooding (LRU)

	A	B	C	D	E	A	B	C	D	E
1	A				E				D	
2		B				A				E
3			C				B			
4				D				C		

10 Cache Misses

Fixing Sequential Flooding (MRU)

	A	B	C	D	E	A	B	C	D	E
1	A									
2		B								
3			C						D	
4				D	E					

4 Hits, 6 Misses

Practice Problem

5. Consider two tables:
Students(sid, name, year, department), 200 pages, 1,000 tuples
Enrolled(sid, course, grade), 500 pages, 6,000 tuples

Query: for each student, list all his/her class grades:

```
SELECT name, course, grade
FROM Students, Enrolled
WHERE Students.sid = Enrolled.sid
```

Remember the Join example we did earlier? We basically did for loops.

Assume that we only have 1 disk, and that we do **not** have to write the resultant tuples back to disk. Consider the join of Student and Enrolled in a nested loop (the naïve nested loops algorithm) with Student as the outer.
Also assume that we **don't** cache any pages in our buffer pool.

1. What is the total number of I/Os this join will require?
2. Of the total number of I/Os, how many are **sequential** I/Os? (Assume that the data for each relation is located in a continuous clump, but the two relations are located in different places.)
   Of the total number of I/Os, how many are **random** I/Os?

Can think of it in terms of:

```
for each student tuple in Students:
        for each enrolled tuple in Enrolled:
                Check if student.sid = enrolled.sid
```

Can think of it in terms of... (Naive)

Students(sid, name, year, department), 200 pages, 1,000 tuples
Enrolled(sid, course, grade), 500 pages, 6,000 tuples

for each student page in Students: (executes 200 times)
    Load student page (1 I/O cost)
    for each student tuple in student page: (1000 total tuples)
        for each enrolled page in Enrolled: (executes 500 times for each student)
            Load enrolled page (1 1/O cost)
            for each enrolled tuple in enrolled page:
                check student.sid == enrolled.sid

Can think of it in terms of... (Slightly Optimized)

Students(sid, name, year, department), 200 pages, 1,000 tuples
Enrolled(sid, course, grade), 500 pages, 6,000 tuples

for each student page in Students: (executes 200 times)
    Load student page (1 I/O cost)
    for each enrolled page in Enrolled: (executes 500 times for each student)
        Load enrolled page (1 1/O cost)
        for each student tuple in student page:
            for each enrolled tuple in enrolled page:
                check student.sid == enrolled.sid

Upcoming Project

Get excited for Project 1: Designing a database with Fields, Tuples, and Buffer Pool

Should be released around this Wednesday!