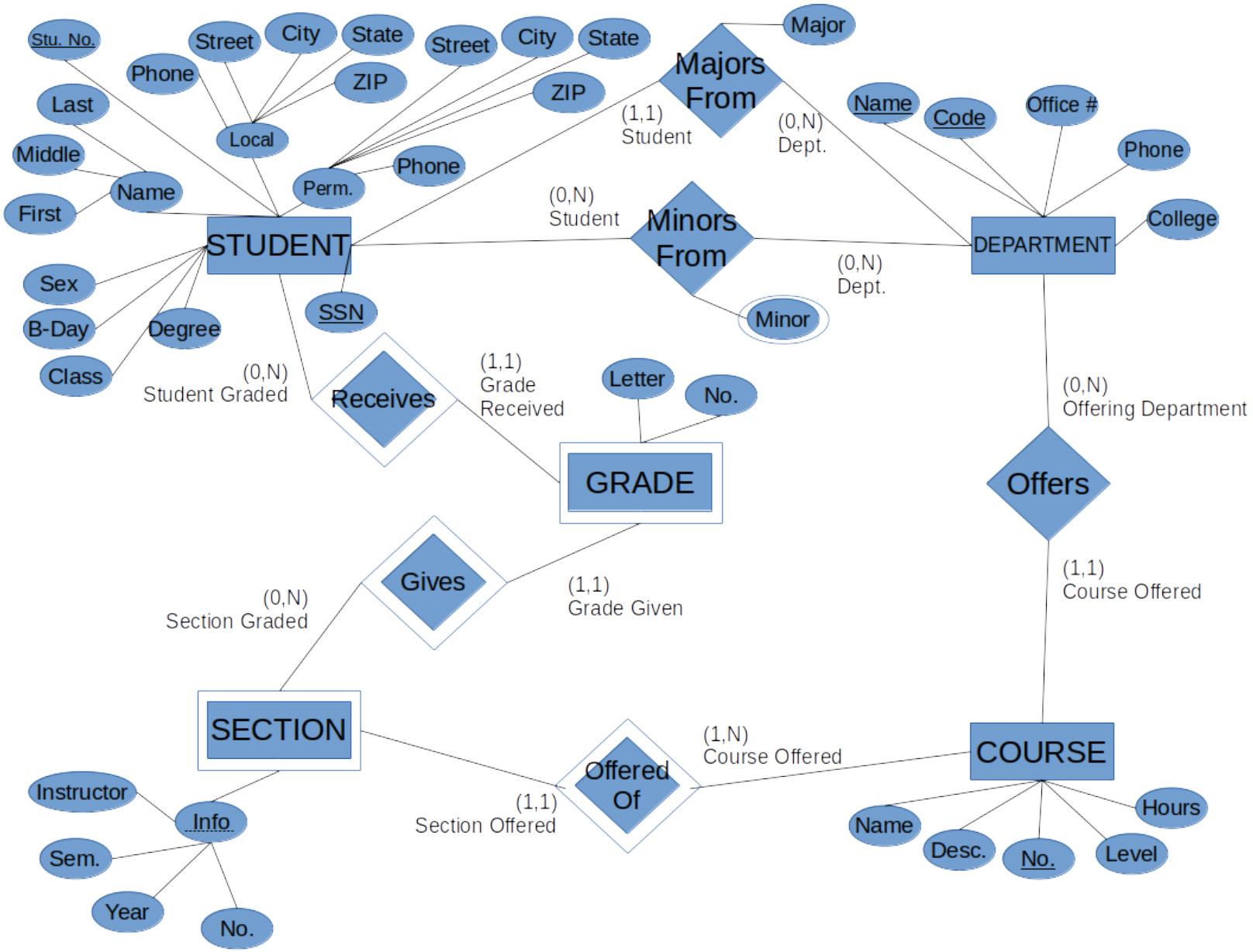
CS 430: Assignment 5

Brandon Ingli

Question 1

ER Diagram



Tables

NOTE: Any table with a ... indicates the table continues on the next line. This is so it fits on the page properly.

1NF

SSN	Student_no	Fname	Minit	Lname	Sex	Bday	Class	Degree	...

...	Local_address	Local_city	Local_state	Local_zip	Local_phone	...

...	Perm_address	Perm_city	Perm_state	Perm_zip	Perm_phone	...

...	Major	Minor	...

...	Major_dept_code	Major_dept_name	Major_dept_office_no	Major_dept_phone	Major_dept_college	...

...	Minor_dept_code	Minor_dept_name	Minor_dept_office_no	Minor_dept_phone	Minor_dept_college	...

...	Course_dept_code	Course_dept_name	Course_dept_office_no	Course_dept_phone	Course_dept_college	...

...	Course_name	Course_no	Course_desc	Course_level	Course_hours	...

...	Section_no	Section_sem	Section_yr	Section_inst	...

...	Grade_ltr	Grade_no

2NF

Student_and_Major_Department

SSN	Student_no	Fname	Minit	Lname	Sex	Bday	Class	Degree	...

...	Local_address	Local_city	Local_state	Local_zip	Local_phone	...

...	Perm_address	Perm_city	Perm_state	Perm_zip	Perm_phone	Major	...

...	Major_dept_code	Major_dept_name	Major_dept_office_no	Major_dept_phone	Major_dept_college

Minor_Department

Minor	Minor_dept_code	Minor_dept_name	...

...	Minor_dept_office_no	Minor_dept_phone	Minor_dept_college

Student_Minors

SSN	Minor

Course

Course_no	Course_name	Course_desc	Course_level	Course_hours	...

...	Course_dept_code	Course_dept_name	Course_dept_office_no	Course_dept_phone	Course_dept_college

Section

Course_no	Section_no	Section_sem	Section_yr	Section_inst

Grade

SSN	Course_no	Section_no	Section_sem	Section_yr	Section_inst	Grade_ltr	Grade_no

3NF

Student

SSN	Student_no	Fname	Minit	Lname	Sex	Bday	Class	Degree	...

...	Local_address	Local_city	Local_state	Local_zip	Local_phone	...

...	Perm_address	Perm_city	Perm_state	Perm_zip	Perm_phone	Major

Majors

Major	Dept_code

Minors

Minor	Dept_code

Departments

Dept_code	Dept_name	Dept_office_no	Dept_phone	Dept_college

Student_Minors

SSN	Minor

Course

Course_no	Course_name	Course_desc	Course_level	Course_hours	Dept_code

Section

Course_no	Section_no	Section_sem	Section_yr	Section_inst

Grade

SSN	Course_no	Section_no	Section_sem	Section_yr	Section_inst	Grade_ltr

Grade_Values

Grade_ltr	Grade_no

FDs and Closures

FDs

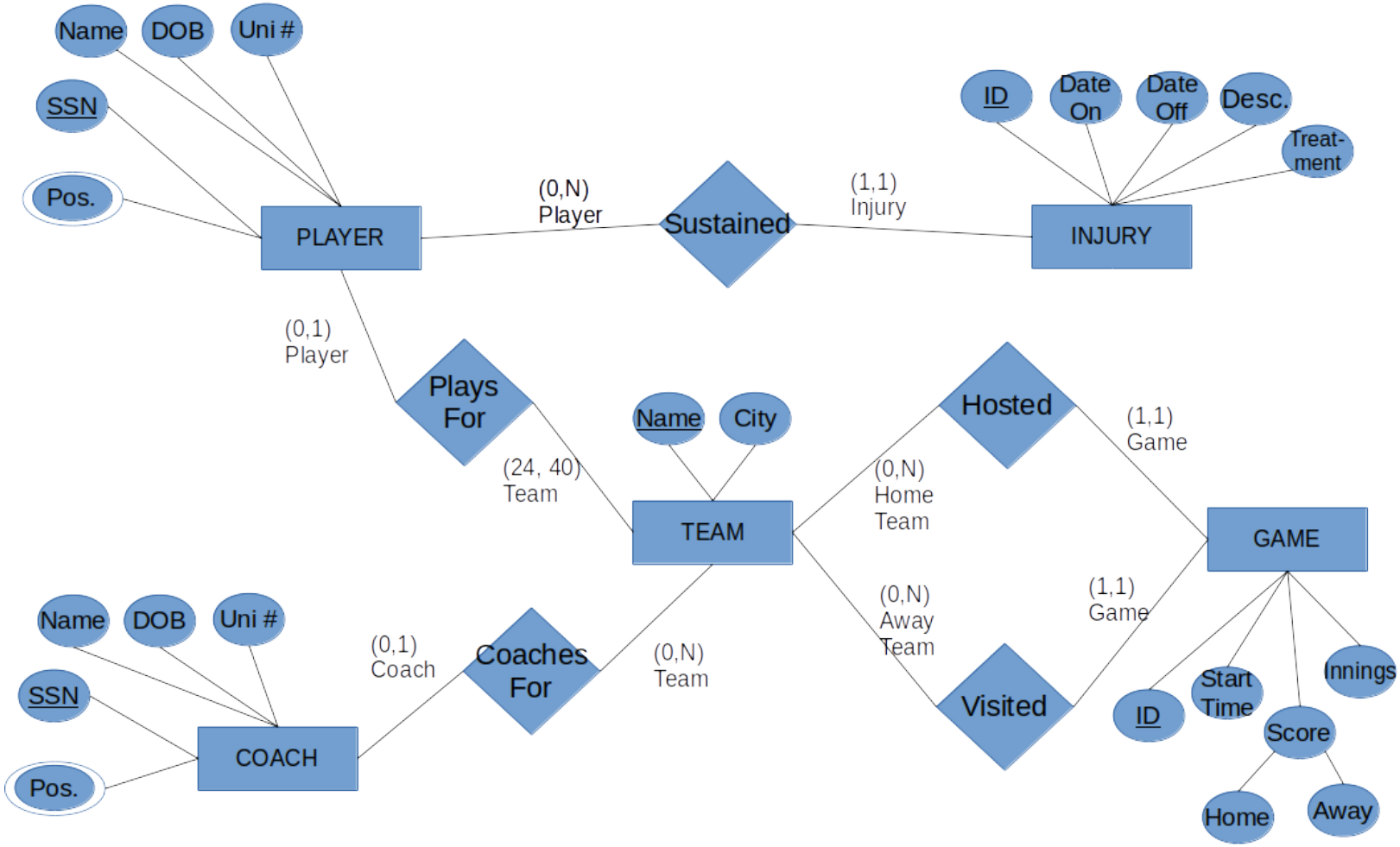
- SSN → Major
- SSN → (Student Demographic Information)
- SSN → Student_no
- Major → {Dept_code, Dept_name}
- Minor → {Dept_code, Dept_name}
- Dept_name → Dept_code
- Dept_code → {Dept_name, Dept_office_no, Dept_phone, Dept_college}
- Course_no → {Course_name, Course_desc, Course_level, Course_hours, Course_dept_code}
- {Course_no, Section_*, SSN} → Grade_ltr
- Grade_ltr → Grade_no

Closures

- SSN → Major_dept_*
- Student_no → Major
- Student_no → (Student Demographic Information)
- Dept_name → {Dept_code, Dept_office_no, Dept_phone, Dept_college}
- Major → {Dept_code, Dept_name, Dept_office_no, Dept_phone, Dept_college}
- Minor → {Dept_code, Dept_name, Dept_office_no, Dept_phone, Dept_college}
- Course_no → {Dept_code, Dept_name, Dept_office_no, Dept_phone, Dept_college}
- {Course_no, Section_*, SSN} → Grade_no
- {Course_no, Section_*, Student_no} → Grade_ltr
- {Course_no, Section_*, Student_no} → Grade_no

Question 2

ER Diagram



Tables

1NF

I will assume that, for clarity, any free-agent player or coach will be listed as playing for the team "Free Agent" instead of having a NULL value.

Player_SSN	Player_name	Player_pos	Player_no	Player_DOB	Player_team_name	Player_team_city	...

...	Coach_SSN	Coach_name	Coach_pos	Coach_no	Coach_DOB	Coach_team_name	Coach_team_city	...

...	Injury_ID	Injury_date_on	Injury_date_off	Injury_desc	Injury_treatment	...

...	Game_ID	Home_team_name	Home_team_city	Away_team_name	Away_team_city	...

...	Game_timestamp	Game_innings	Home_score	Away_score

2NF

Player

Player_SSN	Player_name	Player_DOB	Player_no	Player_team_name	Player_team_city

Player_pos

Player_SSN	Player_pos

Coach

Coach_SSN	Coach_name	Coach_DOB	Coach_no	Coach_team_name	Coach_team_city

Coach_pos

Coach_SSN	Coach_pos

Injury

Injury_ID	Injured_player_SSN	Date_on	Date_off	Desc	Treatment	...

...	Injured_player_name	Injured_player_DOB	Injured_player_no	Injured_player_team_name

Game

Game_ID	Game_timestamp	Game_innings	Home_team_name	Home_team_score	...

...	Away_team_name	Away_team_score	Home_team_city	Away_team_city

Player_in_game

Player_SSN	Game_ID	Player_pos

Coach_in_game

Coach_SSN	Game_ID	Coach_pos

3NF

Team

Team_name	Team_city

Player

Player_SSN	Player_name	Player_DOB	Player_no	Player_team_name

Player_pos

Player_SSN	Player_pos

Coach

Coach_SSN	Coach_name	Coach_DOB	Coach_no	Coach_team_name

Coach_pos

Coach_SSN	Coach_pos

Injury

Injury_ID	Injured_player_SSN	Date_on	Date_off	Desc	Treatment

Game

Game_ID	Game_timestamp	Game_innings	Home_team_name	Home_team_score	...

...	Away_team_name	Away_team_score

Player_in_game

Player_SSN	Game_ID	Player_pos

Coach_in_game

Coach_SSN	Game_ID	Coach_pos

FDs and Closures

FDs

- Player_SSN → {Player_name, Player_DOB, Player_no, Player_team_name}

- Coach_SSN → {Coach_name, Coach_DOB, Coach_no, Coach_team_name}
- Team_name → Team_city
- Injury_ID → {Date_on, Date_off, Desc, Treatment, Player_SSN}
- Game_ID → {Game_timestamp, Game_innings, Home_team_name, Home_team_score, Away_team_name, Away_team_score}

Closures

- Player_SSN → Player_team_city
- Coach_SSN → Coach_team_city
- Injury_ID → {Player_name, Player_DOB, Player_no, Player_team_name}
- Game_ID → {Home_team_city, Away_team_city}

Question 3

$bfr = \lfloor (B/R) \rfloor = \lfloor (1024 \text{ bytes}/400 \text{ bytes}) \rfloor = \lfloor 2.56 \rfloor = 2$

$b = \lceil (r/bfr) \rceil = \lceil (3,000,000/2) \rceil = 1,500,000$

Part a

A linear search of the data would require on average $1,500,000/2 = 750,000$ block accesses.

Part b

A binary search of the data would require approximately $\lceil \lg b \rceil = \lceil \lg 1,500,000 \rceil = \lceil 20.5165 \rceil = 21$ block accesses.

Part c

$r_i = 3,000,000$ since this is a dense index.

$R_i = (9 + 8) = 17$ bytes

$bfr_i = \lfloor (B/R_i) \rfloor = \lfloor (1024 \text{ bytes}/17 \text{ bytes}) \rfloor = \lfloor 60.2353 \rfloor = 60$

$b_i = \lceil (r_i/bfr_i) \rceil = \lceil (3,000,000/60) \rceil = \lceil 50,000 \rceil = 50,000$

A binary search of the index file would require approximately $\lceil \lg b_i \rceil = \lceil \lg 50,000 \rceil = \lceil 15.6096 \rceil = 16$ block accesses, so including the block access to actually retrieve the record, 17 block accesses would be required.

Part d

Using indexes is almost always better because, in most cases, it reduces the number of sectors you need to search through to find the requested record, due to the usually smaller size and sometimes smaller number of indexes. This results in faster record retrieval speeds due to the smaller amount of mechanical drive motion required.

Question 4

Part a

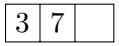
B trees can store data in any node, while B^+ trees can only store data in leaf nodes. Additionally, the leaf nodes are linked together in a B^+ tree, while they are not linked in a B tree.

Part b: Insertion and Deletion

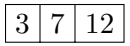
I will assume a B^+ tree with $p = 3$ and $p_{\text{leaf}} = 3$

Insert 7

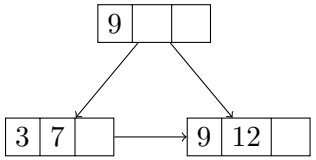
Insert 3



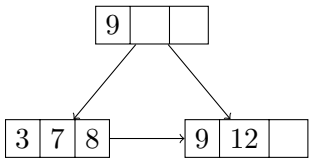
Insert 12



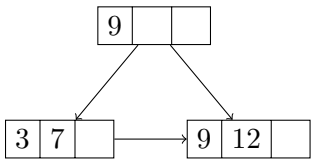
Insert 9



Insert 8



Delete 8



Delete 3