CSE 180: Lab Section 1

- Overview of Lab 1 Assignment
- CREATE TABLE
- Postgres Datatypes
- Keys
- SELECT statement

Tuesday, October 17th
Utkarsh Gupta

Lab Assignment 1: Overview

- Create 7 tables
- Make sure that you are doing it correct:
  - Table names
  - Attribute name and order
  - Data Types
  - Primary Keys
  - Referential Integrity Constraints
- Create the table in the order that they are given

Lab Assignment 1: Overview

SubscriptionKinds(subscriptionMode, subscriptionInterval, rate, stillOffered)

Editions(editionDate, numArticles, numPages)

Subscribers(subscriberPhone, subscriberName, subscriberAddress)

Subscriptions(subscriberPhone, subscriptionStartDate, subscriptionMode, subscriptionInterval, paymentReceived)

Holds(subscriberPhone, subscriptionStartDate, holdStartDate, holdEndDate)

Articles(editionDate, articleNum, articleAuthor, articlePage)

ReadArticles(subscriberPhone, editionDate, articleNum, readInterval)

Lab Assignment 1: Overview (cont)

- Workflow:

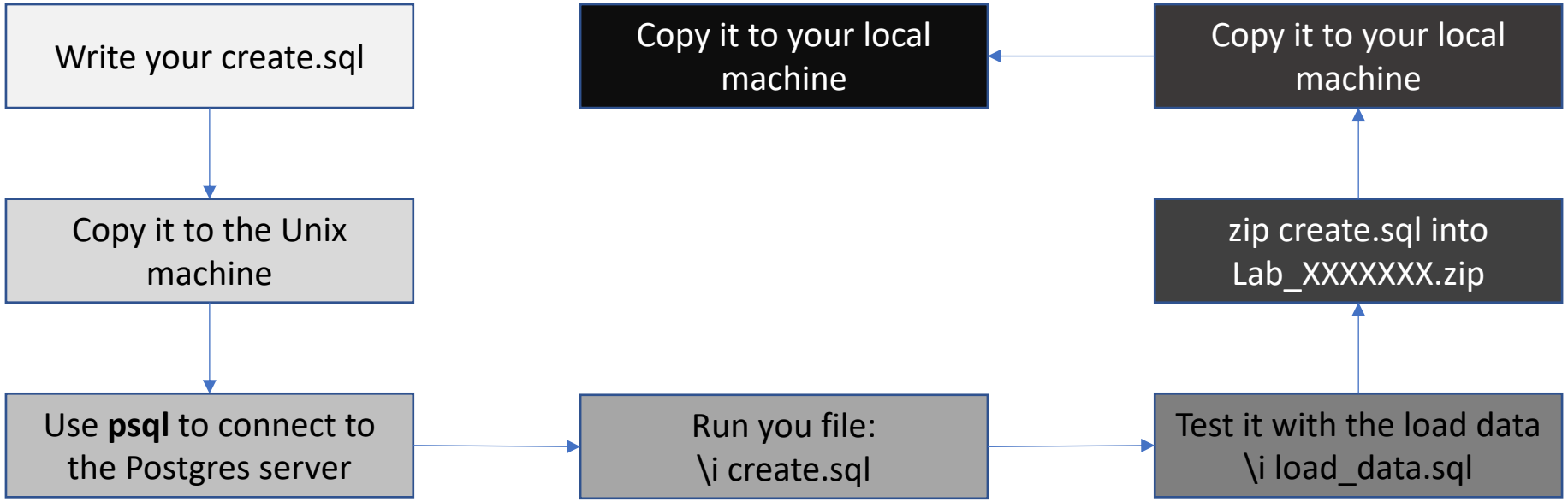

Diagram taken from Tanuj's slides

CREATE TABLE

General syntax:

```
CREATE TABLE TABLE_NAME (
        COLUMN_1   DATA_TYPE,
        COLUMN_2   DATA_TYPE,
        COLUMN_3   DATA_TYPE,
        KEY_CONSTRAINTS
);
```

Example

```
CREATE TABLE STUDENT (
        S_ID INT,
        S_NAME VARCHAR(30),
        SSN CHAR(9),
        PHONE CHAR(10),
        EMAIL VARCHAR(50),
)
```

Postgres Datatypes

- **Numeric types:**
  - **INTEGER [INT]**: Signed 4 bytes integer
  - **Numeric (p, s) [DECIMAL (p, s)]** : Selectable precision numeric

    p = Total digits

    s = Position of the decimal point

    Example:

    NUMERIC (2, 2):

    -> Will only allow you to store numbers like 0.1, 0.01, 0.001 as 0.00

    -> Can't store 1.3, Why?

    NUMERIC (4, 0):

    -> You can store numbers: 1, 10, 100, 1000, **0.99999999 and 0.09999999**

    -> Numbers like 1.1, 1.111111111111 will get rounded to 1 **(s is 0)**

    -> Can't store number like 10000

    NUMERIC (4, 2):

    -> Can store numbers like 11.11, 1.111111 (rounded to 1.11),

    -> Can you store 111?

Postgres Datatypes (cont.)

- **Character types:**
  - **CHARACTER(n) [CHAR(n)]:**
    - This can store ‘n’ characters
    - Hence, it create space for n characters in advance
    - Faster (static memory allocation) but space inefficient
  - **CHARACTER VARYING(n) [VARCHAR(n)]:**
    - This can store ‘n’ characters
    - But this **DOES NOT** create space for ‘n’ characters upfront
    - Only creates space w.r.t. to the length of the values that you will insert
    - Slower (dynamic memory allocation) but space efficient

Postgres Datatypes (cont.)

- **Date**
  - Calendar date with format YYYY-MM-DD (year, month, day), e.g. 2023-01-13
- **Boolean [BOOL]**
  - TRUE or FALSE

Postgres Datatypes (cont.)

Example of Datatypes in CREATE TABLE:

```
CREATE TABLE STUDENT (
        S_ID INT,
        S_NAME VARCHAR(30),
        SSN CHAR(9),
        DOB DATE,
        GPA NUMERIC(3, 2),
        HAS_GRADUATED BOOL,
        PHONE CHAR(10),
        PRIMARY KEY (S_ID)
);
```

Keys

What are keys?

The concept of keys helps us to find a unique row in the table.

Need for having keys:

S_ID	S_NAME	SSN	DOB	GPA	HAS_GRAD	PHONE
1	**DAVID**	1234	1997-01-01	3.8	**F**	**0123456**
2	**JULIA**	4567	2000-02-18	3.9	**F**	**1234589**
3	**DAVID**	2468	2000-02-01	3.8	**F**	**9827123**

Suppose, you want to set the **HAS_GRAD** value of student **DAVID** to True.

Keys (cont.)

Super Key

It is the super set of the possible attributes that can help you to uniquely identify a row in the table.

Example:

{S_ID}, {SSN}, {S_ID, S_NAME}, {S_ID, SSN}, {S_NAME, SSN}, {S_ID, S_NAME, SSN}, {S_NAME, PHONE}, … etc.

S_ID	S_NAME	SSN	DOB	GPA	HAS_GRAD	PHONE
1	**DAVID**	1234	1997-01-01	3.8	**F**	**0123456**
2	**JULIA**	4567	2000-02-18	3.9	**F**	**1234589**
3	**DAVID**	2468	2000-02-01	3.8	**F**	**9827123**

Keys (cont.)

Candidate Key or just key

Minimal set of attributes that can help you identify a unique row in the table.

Example:

Key	Super	Candidate
{S_ID}	YES	YES
{SSN}	YES	YES
{S_ID, S_NAME}	YES	NO
{S_ID, SSN}	YES	NO
{S_NAME, SSN}	YES	NO
{S_ID, S_NAME, SSN}	YES	NO
{S_NAME}	NO	NO
{S_NAME, PHONE}	YES	YES

Keys (cont.)

Primary Key

Primary key helps us to uniquely identify a row in the table.

A chosen **CANDIDATE KEY** becomes the primary key.

It can either be a single attribute or a collection of attributes:

Example: **{S_ID}, {SSN}, {S_NAME, PHONE}**

S_ID	S_NAME	SSN	DOB	GPA	HAS_GRAD	PHONE
1	**DAVID**	1234	1997-01-01	3.8	**F**	**0123456**
2	**JULIA**	4567	2000-02-18	3.9	**F**	**1234589**
3	**DAVID**	2468	2000-02-01	3.8	**F**	**9827123**

- Every table should have a primary key.
- A table can only have at most one primary key.

Keys (cont.)

Primary Key (cont.)

```
CREATE TABLE STUDENT (
        S_ID INT PRIMARY KEY,
        S_NAME VARCHAR(30),
        SSN CHAR(9),
        DOB DATE,
        GPA NUMERIC(3, 2),
        HAS_GRADUATED BOOL,
        PHONE CHAR (10)
);
```

```
CREATE TABLE BEERS (
        BEER VARCHAR (30) PRIMARY KEY,
        MANF VARCHAR(50)
);
```

```
CREATE TABLE BARS (
        BAR VARCHAR (30),
        LICENSE VARCHAR(50),
        PRIMARY KEY (BAR, LICENSE)
);
```

Keys (cont.)

Alternate Key

The candidate keys which were not chosen as a Primary key are the Alternate keys

Example: **{SSN}, {S_NAME, PHONE}**

S_ID	S_NAME	SSN	DOB	GPA	HAS_GRAD	PHONE
1	**DAVID**	1234	1997-01-01	3.8	**F**	**0123456**
2	**JULIA**	4567	2000-02-18	3.9	**F**	**1234589**
3	**DAVID**	2468	2000-02-01	3.8	**F**	**9827123**

Keys (cont.)

Foreign Key

Foreign key specify that values in a column (or a group of columns) must match values in column of another/same table.

It maintains Referential Integrity between tables.

STUDENTS TABLE Example: A student enrolled in CSE_180 must exist in the Student table.

S_ID	S_NAME	SSN	DOB	GPA	HAS_GRAD	PHONE
1	**DAVID**	1234	1997-01-01	3.8	**F**	**0123456**
2	**JULIA**	4567	2000-02-18	3.9	**F**	**1234589**
3	**DAVID**	2468	2000-02-01	3.8	**F**	**9827123**

ENROLMENT TABLE

S_ID	COURSE_NUM	CREDITS
1 ✓	CSE_180	5
5 ✗	CSE_180	5
3 ✓	CSE_181	2

Keys (cont.)

Foreign Key (cont.)

```
CREATE TABLE ENROLMENT (
    S_ID INT,
    COURSE_NUM VARCHAR(30),
    PRIMARY KEY (S_ID, COURSE_NUM),
    FOREIGN KEY (S_ID) REFERENCES
    STUDENTS
```

STUDENTS TABLE

S_ID	S_NAME	SSN	DOB	GPA	HAS_GRAD	PHONE
1	**DAVID**	1234	1997-01-01	3.8	**F**	**0123456**
2	**JULIA**	4567	2000-02-18	3.9	**F**	**1234589**
3	**DAVID**	2468	2000-02-01	3.8	**F**	**9827123**

ENROLMENT TABLE

S_ID	COURSE_NUM	CREDITS
1	CSE_180	5
3	CSE_181	2

Keys (cont.)

Foreign Key (cont.)

```
CREATE TABLE RANK_HOLDERS (
   STUDENT_ID INT,
   COURSE VARCHAR(30)
   RANK INT,
   PRIMARY KEY (STUDENT_ID, COURSE),
   FOREIGN KEY (STUDENT_ID, COURSE) REFERENCES ENROLMENT (S_ID, COURSE_NUM)
);
```

S_ID	COURSE_NUM	CREDITS
1	CSE_180	5
1	CSE_181	2
3	CSE_181	2
3	CSE_180	5

ENROLMENT TABLE

STUDENT_ID	COURSE	RANK
1	CSE_180	1
1	CSE_181	2
3	CSE_180	2
3	CSE_181	1

RANK_HOLDERS TABLE

SELECT Statement

SELECT statement:

1. Retrieves rows from one or more tables
2. All tables in the FROM clause are used to get the result
3. Conditions under WHERE clause are used to filter the data
4. Other operators such as GROUP BY, ORDER BY, HAVING will be explored later in the course

SELECT Statement (cont.)

SELECT statement (cont.):

Example: SELECT * FROM STUDENT;

*** :** Get me all the attributes

Examples:

SELECT * FROM ENROLMENT WHERE COURSE_NUM = ‘CSE_180’;

SELECT * FROM SELLS WHERE PRICE > 3;

SELECT BEER FROM SELLS WHERE PRICE < 2 AND BAR=‘Front & Cooper’;

Tips for Lab 1

1. Compare with Beer Scripts posted on Piazza.
2. Check the lengths of data types like char and varchar.
3. Check the meaning of **p, s** and in the NUMERIC data type.
4. Get help from the TAs as soon as you can.
5. Do not share solutions or screenshots of any SQL statement through Piazza, discord or any other medium except the TAs.