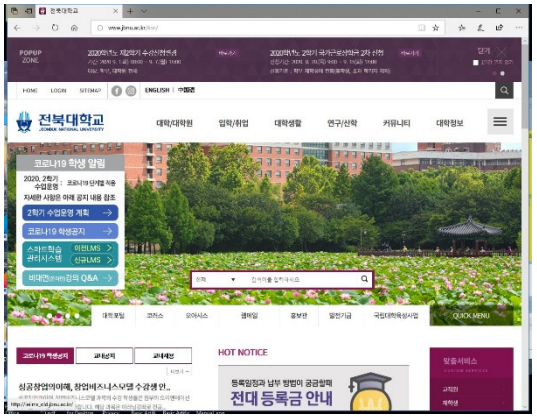
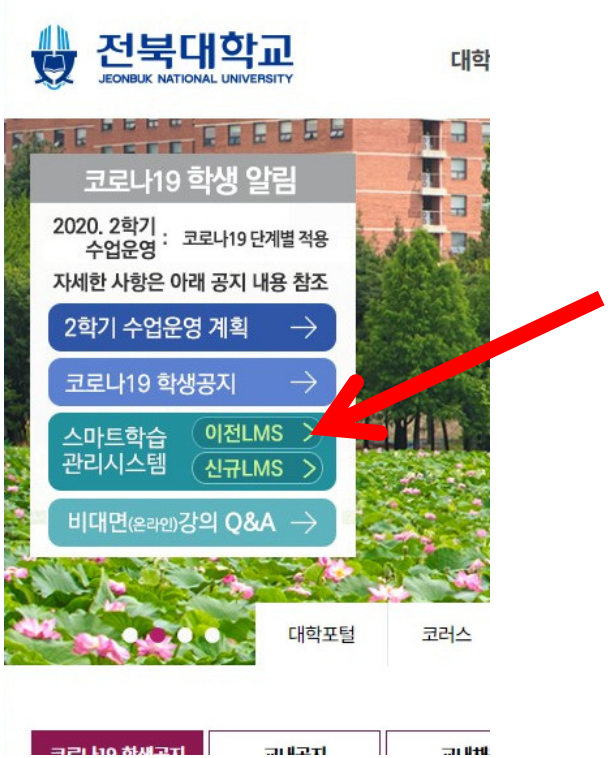
Introduction to Data Structure (Data Management)

Felipe P. Vista IV

Announcement



http://jbnu.ac.kr/



http://ieilms_old.jbnu.ac.kr/ → http://ieilmsold.jbnu.ac.kr/

Reminder

- Everybody, make sure that your name in ZOOM is in the following format:
  - University ID Num Name
  - Ex: 202054321 Juan Dela Cruz

  - Not changing your name, you might be marked Absent
  *

Course Outline

- Introduction, Data Models, SQL Basics
- SQL Aggregates, Grouping, Subqueries
- Wrapping-up SQL, Relational Algebra (RA), Datalog
- NoSQL, JSON
- JSON, SQL++
- SQL++, RA Part II, Query Evaluation
- Storage, Indexing Basics
- Basics of Query Optimization, Parallel Databases
- Map Reduce, Spark
- E/R Diagrams, Constraints
- Design Theory
- Transactions
- DB Techniques for Machine Learning

Data Management

SQL BASICS (MORE OF IT)

Multi-column Keys

- Creating/naming a key

```
CREATE TABLE Crew(
        name VARCHAR(20) PRIMARY KEY,
        country VARCHAR(20),
        occupation VARCHAR(40),
        years_biking INT,
        has_bike INT);
```

- How about if we want the columns "`name`" & "`country`" to form the key?

Multi-column Keys

- Creating/naming a key

```
CREATE TABLE Crew(
        name VARCHAR(20) PRIMARY KEY,
        country VARCHAR(20),
        occupation VARCHAR(40),
        years_biking INT,
        has_bike INT);
```

PRIMARY KEY(name));

- How about if we want the columns "name" & "country" to form the key?

Multi-column Keys

- We will update the syntax* to reflect primary key creation with two columns

* syntax – rules to correctly structure statements or expressions for a particular language

Multi-column Keys

- We will update the syntax* to reflect primary key creation with two columns

```
CREATE TABLE Crew(
     name VARCHAR(20) PRIMARY KEY,
     country VARCHAR(20),
     occupation VARCHAR(40),
     years_biking INT,
     has_bike INT,
     PRIMARY KEY (name, country));
```

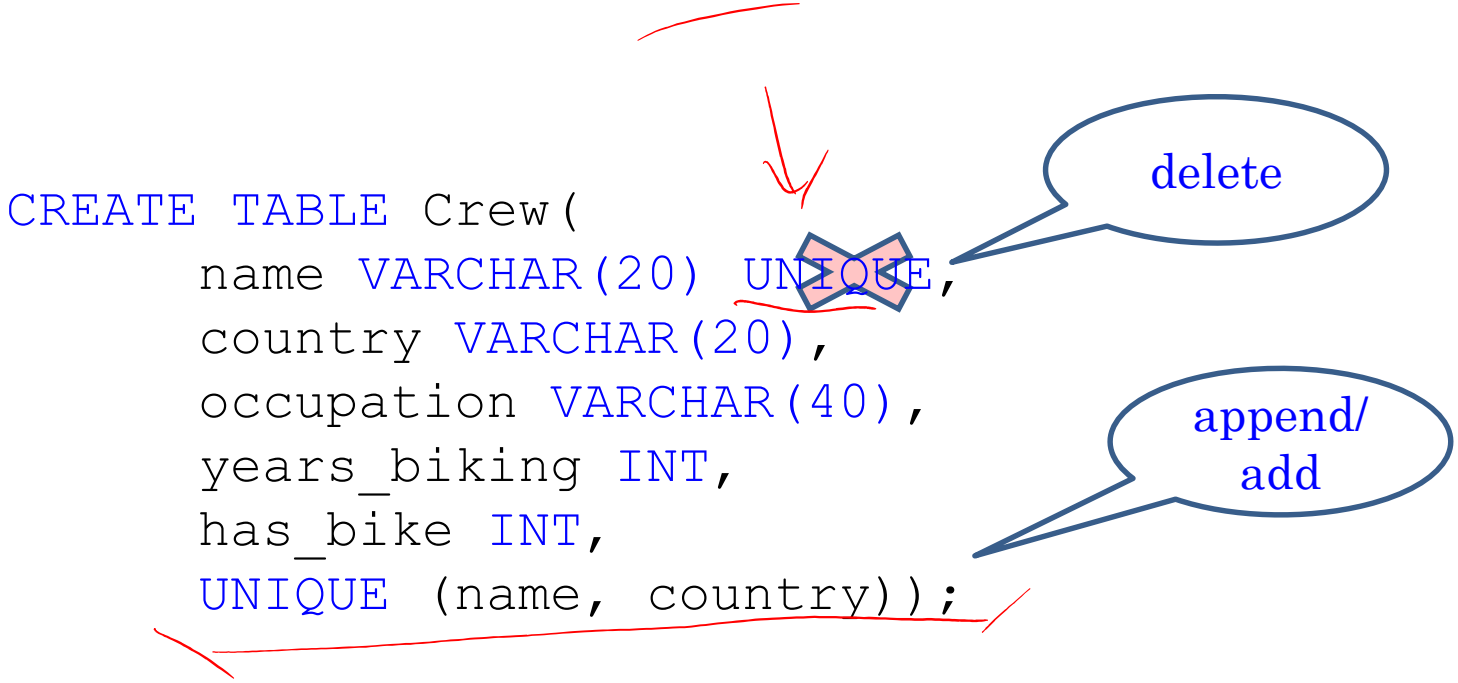
delete

append/ add

* syntax – rules to correctly structure statements or expressions for a particular language

Multi-column Keys

- The same applies for secondary keys



* syntax – rules to correctly structure statements or expressions for a particular language

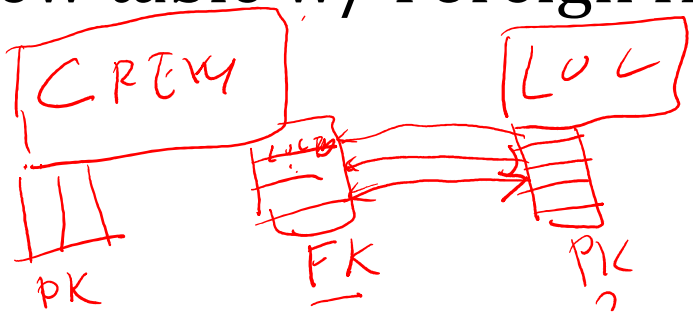
Multi-column Keys

- To set LocID as foreign key, remember Location table, we first set LocID as Primary Key, then use it

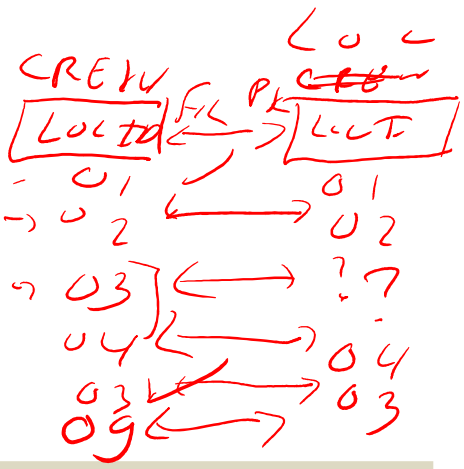
```
CREATE TABLE location(
    LocID INT PRIMARY KEY, dong VARCHAR(40),
    city VARCHAR(40), province VARCHAR(40));
```

Multi-column Keys

- We drop then re-create the Crew table w/ Foreign Key




```
DROP TABLE Crew;
CREATE TABLE Crew(
        name VARCHAR(20), country VARCHAR(20),
        occupation VARCHAR(40),
        years_biking INT, has_bike INT,
        locID INT,
        FOREIGN KEY (locID)
           REFERENCES location(locID),
        PRIMARY KEY (name, country));
```



Multi-column Keys

- Another syntax to set Foreign Key for same tables

```
DROP TABLE Crew;
CREATE TABLE Crew(
      name VARCHAR(20), country VARCHAR(20),
      occupation VARCHAR(40),
      years_biking INT, has_bike INT,
      location INT
         REFERENCES location(LocID),
      PRIMARY KEY (name, country));
```

Multi-column Keys

- In case of multi-column keys

```
DROP TABLE Crew;
CREATE TABLE Crew(
        name VARCHAR(20), country VARCHAR(20),
        occupation VARCHAR(40),
        years_biking INT, has_bike INT,
        locID INT, zipCode INT,
        FOREIGN KEY (locID, zipCode)
           REFERENCES location(LocID, zipCode),
        PRIMARY KEY (name, country));
```

Multi-column Keys

- Another syntax to set Foreign Key for same tables

```
DROP TABLE Crew;
CREATE TABLE Crew(
      name VARCHAR(20), country VARCHAR(20),
      occupation VARCHAR(40),
      years_biking INT, has_bike INT,
      location INT, zipCode INT
         REFERENCES location(LocID, zipCode),
      PRIMARY KEY (name, country));
```

A way to input data

- Write a program that will output SQL statements

```
for (int x=1; x<=24; x++)
   for (int y=1; y<=43; y++)
      System.out.format(
        "INSERT INTO numTable VALUES (%d,%d);\n,
```

A way to input data

- Write a program that will output SQL statements

```
for (int x=1; x<=24; x++)
   for (int y=1; y<=43; y++)
      System.out.format(
        "INSERT INTO numTable VALUES (%d,%d);\n,
```

- Pipe/feed the program output into SQLite
  - `sqlite3 samp.db < inputs.sql`

A way to input data

- Write a program* that will output SQL statements

```
public class HelloWorld{
     public static void main(String []args){

         for (int x=1;x<=24;x++)
             for (int y=1;y<=1;y++)
                 System.out.format (
                     "INSERT INTO numTable VALUES (%d, %d);\n", x,y);
     }
}
```

- Pipe/feed the program output into SQLite
  - `sqlite3 samp.db < inputs.sql`

* Online Java Compiler:
https://www.tutorialspoint.com/compile_java_online.php

Warning when writing programs

- Take note when dealing with strings

```
System.out.format(
    "INSERT INTO sampTable VALUES (%d,'%s');",
    2,"O'Hara");
```

Warning when writing programs

- Take note when dealing with strings

```
System.out.format(
    "INSERT INTO sampTable VALUES (%d,'%s');",
    2,"O'Hara");
```

- Output will be

```
INSERT INTO sampTable VALUES 2,'O'Hara';
```

  - and this will give a problem due to syntax error.

Warning when writing programs

- Take note when dealing with strings

```
System.out.format(
    "INSERT INTO sampTable VALUES (%d,'%s');",
    2,"O'Hara");
```

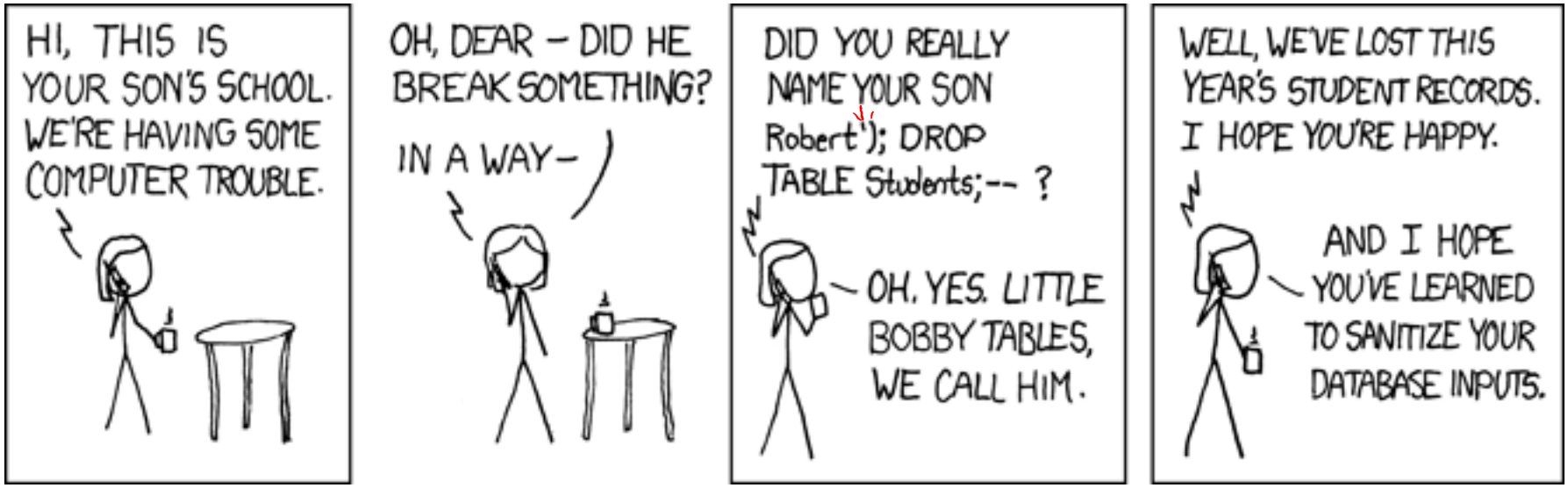
- Output will be

```
INSERT INTO sampTable VALUES 2,'O'Hara');
```

  - and this will give a problem due to syntax error

- not only error, this allows a SQL injection attack!
  - Check for quotes and escape (or disallow them)

Warning when writing programs



https://xkcd.com/327/

Insert into student values ('248...', 'ello', 'pass', Robert'); DROP TABLE STUDENTS; --

Crew D

DISTINCT and ORDER BY

- Query results does not need to have relations
  - can return duplicate rows
  - remove duplicates using DISTINCT

```
SELECT DISTINCT country FROM Crew;
```

OUTPUT

```
country
Iran
South Africa
USA
South Korea
Hong Kong
Philippines
```

DISTINCT and ORDER BY

- Order of results is usually not specified
  - set the order using ORDER BY column ASC/DESC

```
SELECT DISTINCT country FROM Crew
ORDER BY country ASC;
```

OUTPUT

```
country
Hong Kong
Iran
Philippines
South Africa
South Korea
USA
```

Joins

- Can use data from multiple tables
  - selection and demonstration of 'joining' the tables crew & location

```
SELECT name, dong FROM crew, location WHERE
crew.locID=location.locID AND NOT crew.has_bike;
```

OUTPUT

```
name|dong
Matt|Songcheondong
Mikki|Songcheondong
Khan Boy|Inhudong
```

Joins

- Can use data from multiple tables
  - selection and demonstration of 'joining' the tables crew & location

```
SELECT name, dong FROM crew, location WHERE
crew.locID=location.locID;
```

OUTPUT

```
name|dong
Soheil|Deokjindong
Nwabisa|Soshindong
Matt|Songcheondong
Mikki|Songcheondong
Divan|Deokjindong
Khan Boy|Uadong
Pat|Inhudong
Janin|Deokjindong
```

Interpreting Joins

- A JOIN B
  - Produce one row for every pair of rows
  - one row from A and one row from B

crew.LocID = location.crewID

LocID	Dong
1	Songcheondong
2	Soshindong
3	Deokjindong
4	Inhudong
5	Uadong

Name	Has_Bike	LocID
Soheil	1	3
Nwabisa	1	2
Matt	0	1
Mikki	0	1
Divan	1	3
Khan Boy	0	5
Pat	1	4
Janin	1	3

Song

(1, Deokjindong, Matt, 0, 1)

Interpreting Joins

- A JOIN B
  - Produce one row for every pair of rows
  - one row from A and one row from B

LocID	Dong
1	Songcheondong
2	Soshindong
3	Deokjindong
4	Inhudong
5	Uadong

Name	Has_Bike	LocID
Soheil	1	3
Nwabisa	1	2
Matt	0	1
Mikki	0	1
Divan	1	3
Khan Boy	0	5
Pat	1	4
Janin	1	3

(1,Deokjindong,Mikki,0,1)

Interpreting Joins

- A JOIN B
  - Produce one row for every pair of rows
  - one row from A and one row from B

LocID	Dong
1	Songcheondong
2	Soshindong
3	Deokjindong
4	Inhudong
5	Uadong

Name	Has_Bike	LocID
Soheil	1	3
Nwabisa	1	2
Matt	0	1
Mikki	0	1
Divan	1	3
Khan Boy	0	5
Pat	1	4
Janin	1	3

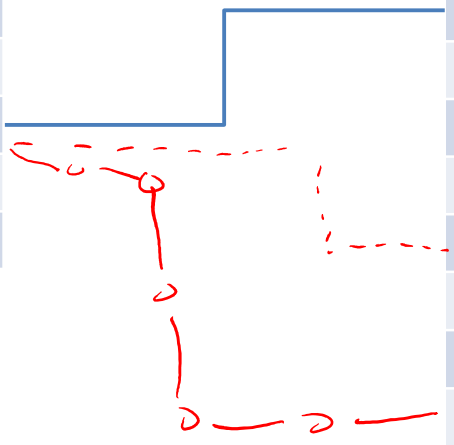
```
(2,Soshindong,Nwabisa,1,2)
```

Interpreting Joins

- A JOIN B
  - Produce one row for every pair of rows
  - one row from A and one row from B

LocID	Dong
1	Songcheondong
2	Soshindong
3	Deokjindong
4	Inhudong
5	Uadong

Name	Has_Bike	LocID
Soheil	1	3
Nwabisa	1	2
Matt	0	1
Mikki	0	1
Divan	1	3
Khan Boy	0	5
Pat	1	4
Janin	1	3



```
(3,Deokjindong,Soheil,1,3)
```

Interpreting Joins

- A JOIN B
  - Produce one row for every pair of rows
  - one row from A and one row from B

LocID	Dong
1	Songcheondong
2	Soshindong
3	Deokjindong
4	Inhudong
5	Uadong

Name	Has_Bike	LocID
Soheil	1	3
Nwabisa	1	2
Matt	0	1
Mikki	0	1
Divan	1	3
Khan Boy	0	5
Pat	1	4
Janin	1	3

```
(3,Deokjindong,Divan,1,3)
```

Interpreting Joins

- A JOIN B
  - Produce one row for every pair of rows
  - one row from A and one row from B

LocID	Dong
1	Songcheondong
2	Soshindong
3	Deokjindong
4	Inhudong
5	Uadong

Name	Has_Bike	LocID
Soheil	1	3
Nwabisa	1	2
Matt	0	1
Mikki	0	1
Divan	1	3
Khan Boy	0	5
Pat	1	4
Janin	1	3

```
(3,Deokjindong, Janin,1,3)
```

Interpreting Joins

- A JOIN B
  - Produce one row for every pair of rows
  - one row from A and one row from B

LocID	Dong
1	Songcheondong
2	Soshindong
3	Deokjindong
4	Inhudong
5	Uadong

Name	Has_Bike	LocID
Soheil	1	3
Nwabisa	1	2
Matt	0	1
Mikki	0	1
Divan	1	3
Khan Boy	0	5
Pat	1	4
Janin	1	3

```
(4,Inhudong,Pat,1,4)
```

Interpreting Joins

- A JOIN B
  - Produce one row for every pair of rows
  - one row from A and one row from B

LocID	Dong
1	Songcheondong
2	Soshindong
3	Deokjindong
4	Inhudong
5	Uadong

Name	Has_Bike	LocID
Soheil	1	3
Nwabisa	1	2
Matt	0	1
Mikki	0	1
Divan	1	3
Khan Boy	0	5
Pat	1	4
Janin	1	3

(5,Uadong,Khan Boy,0,5)

Interpreting Joins

- A JOIN B
  - Produce one row for every pair of rows
  - one row from A and one row from B

LocID	Dong
1	Songcheondong
2	Soshindong
3	Deokjindong
4	Inhudong
5	Uadong

JOIN

Name	Has_Bike	LocID
Soheil	1	3
Nwabisa	1	2
Matt	0	1
Mikki	0	1
Divan	1	3
Khan Boy	0	5
Pat	1	4
Janin	1	3

Interpreting Joins

- A JOIN B
  - Produce one row for every pair of rows
  - one row from A and one row from B

LocID	Dong
1	Songcheondong
2	Soshindong
3	Deokjindong
4	Inhudong
5	Uadong

JOIN

Name	Has_Bike	LocID
Soheil	1	3
Nwabisa	1	2
Matt	0	1
Mikki	0	1
Divan	1	3
Khan Boy	0	5
Pat	1	4
Janin	1	3

- This join produces 8 diff rows
  - Generally: #rows = (#rowsA)(#rowsB)
  - Num of rows usually much smaller after selection…
  - DBMS will do all it can to not compute A JOIN B

Assignment

- Assignment will be posted at the **IEILMS** later

+ 17 people 30%

NL = 100 pts

L = ∠ 90/35

∠ 70/25

NS = 50

Thank you.