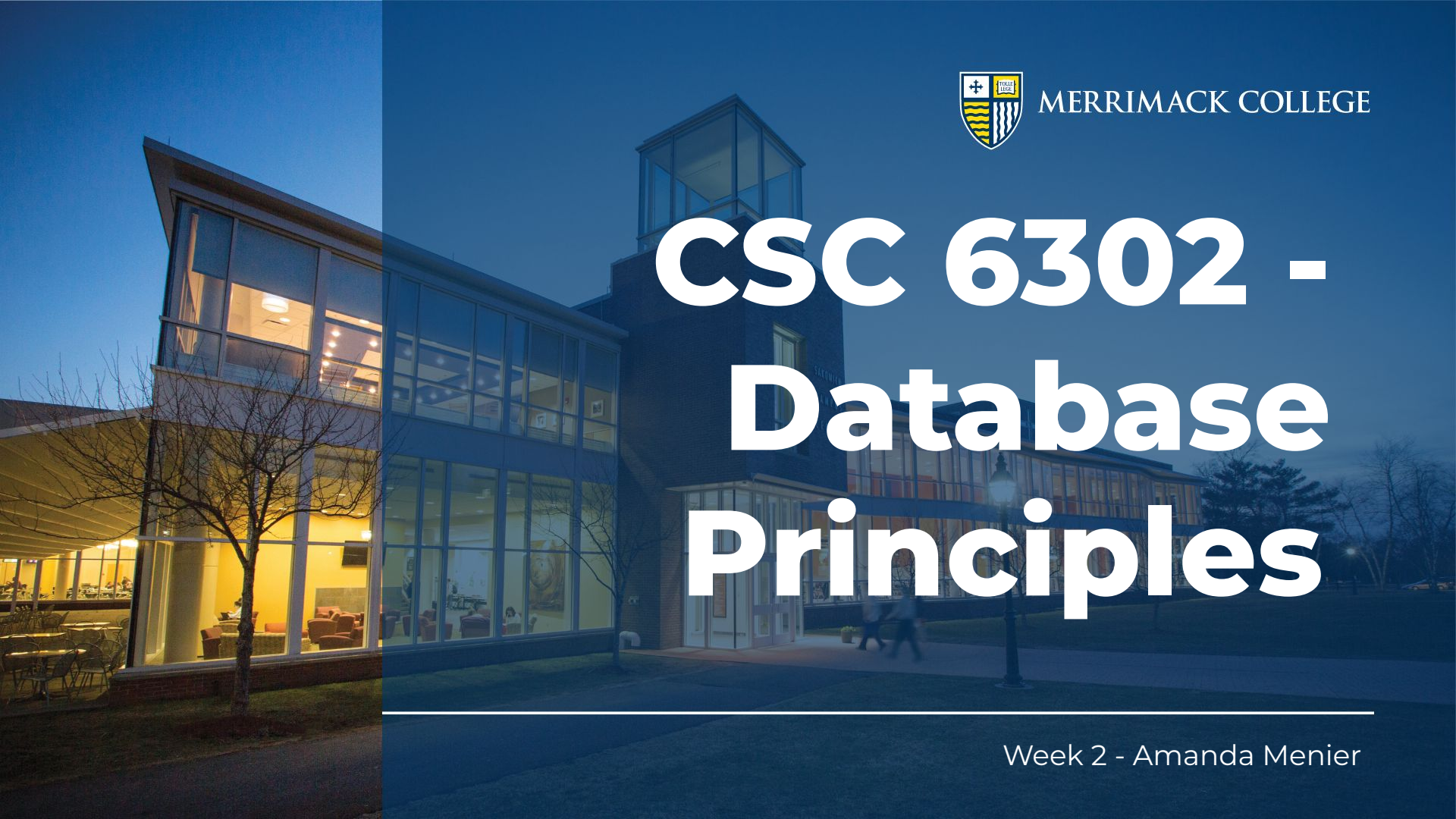
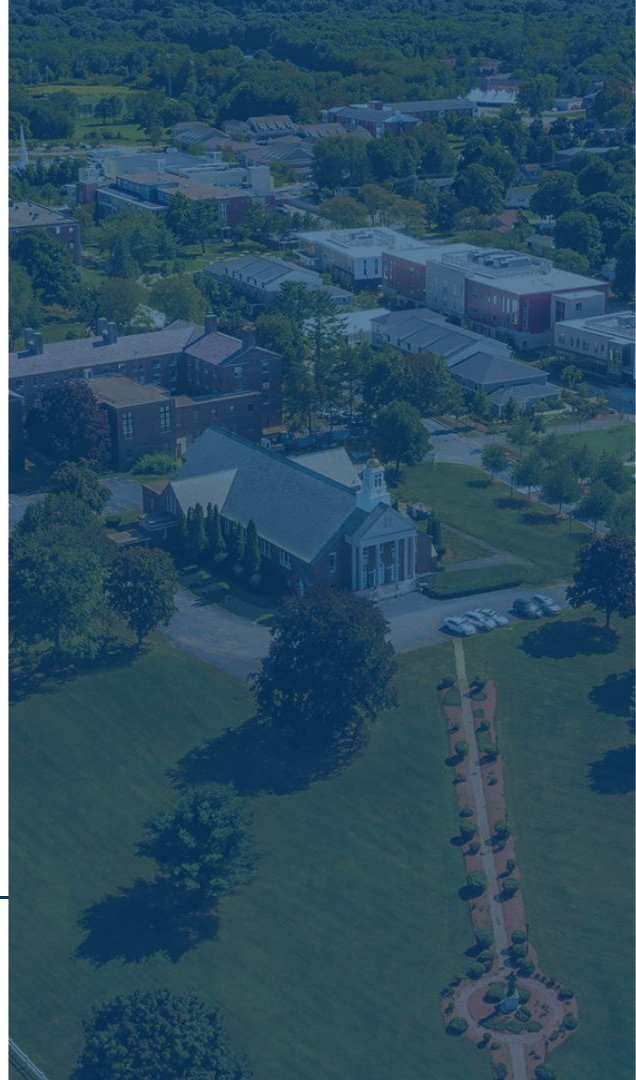

MERRIMACK COLLEGE
CSC 6302 - Database Principles
Week 2 - Amanda Menier

Agenda

I will be away Saturday to Monday and will not be grading until Tuesday

Discussion Topics:

- Information Redundancy
- Primary Keys and Uniqueness
- Decomposition
- Foreign Keys and Relationships
- Normal Forms
- Joins
- Clean and Transform



Project 1 Notes

1) When choosing datatypes think about how we will use the data in the future
   a) If we are going to add/subtract/multiply the data, choose a number type
      i) If we need precision, use DECIMAL
         (1) If using DECIMAL, think about the magnitude of values
   b) If data are categories or identifiers, choose a string/text type
      i) VARCHAR is short for Variable Character - meaning data entered in that field will have different lengths. This is good for names or addresses where we can't predict the length.
      ii) CHARs will have a fixed length
   c) Date and time information should always use the date and time types so we can perform special operations on them.

https://stackoverflow.com/questions/618535/difference-between-decimal-float-and-double-in-net/618596#618596
https://code.rohitink.com/2013/06/12/mysql-integer-float-decimal-data-types-differences/

Project 1 Notes

1. Single column queries
    a. Usually not useful unless we are summing for looking for a single answer
    b. Our queries are questions we are asking to the database - what are useful questions?
    c. The relationships between our data across columns and tables are what make things interesting
        i. Typically not useful if we have the same single value repeated multiple times with no other information
2. Having a ‘drop’ statement without ‘if exists’ will cause an error if the database does not exist when the code is run
3. Careful reading

Repetition & Redundancy

If we must enter repeated information, we risk errors.

Ideally we want one “Source of Truth” for each important piece of information

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

What can go wrong?

Insertion Anomaly:

Unable to insert data. Common causes include required fields that are null, wrong data types.

Deletion Anomaly:

Deleting a record causes errors in related tables.

Update Anomaly:

Same information in multiple places. One is updated but another isn't. Now the data doesn't agree.

https://www.geeksforgeeks.org/anomalies-in-relational-model/

Solution: Decomposition and Normalization

Decomposition:

The process of breaking down a complex database schema into simpler, more manageable sub-schemas.

Normalization:

Creating our new tables following a set of rules about good table design

Solution: Decomposition and Normalization

Decomposition:

The process of breaking down a complex database schema into simpler, more manageable sub-schemas.

Normalization:

Creating our new tables following a set of rules about good relational design.

A database in “good” form has no redundant data and that can be queried quickly, and efficiently.

First Normal Form (1NF)

1. Cannot use row order to convey information
2. Cannot mix data types within the same column
3. Cannot have tables without a primary key
4. Cannot have repeating data groups
    a. Scores column (10, 9, 10, 4, 2, 10)
5. Cannot have columns with the same name
    a. DBMS usually won't let you do this

How Do We Choose Keys?

Superkey:

A set of one or more attributes that collectively allow us to identify uniquely a tuple in the relation

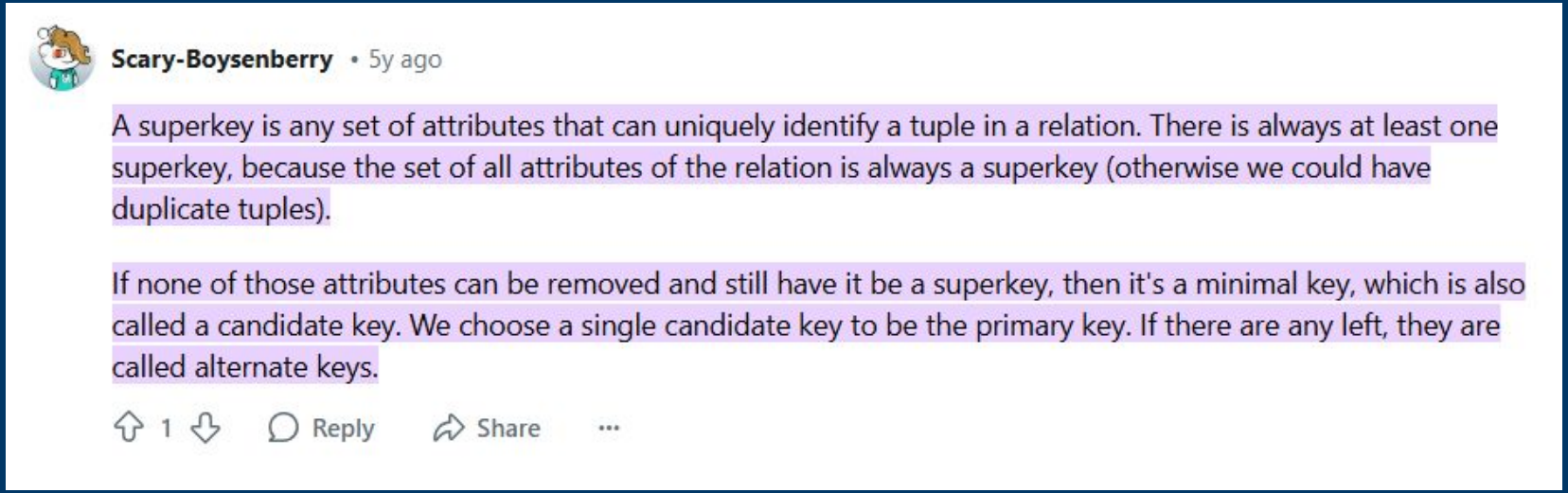
Candidate Key:

A minimal superkey. Requires the fewest attributes to uniquely identify a tuple

Primary Key:

The candidate key that best suits our design

How Do We Choose Keys?



Scary-Boysenberry • 5y ago

A superkey is any set of attributes that can uniquely identify a tuple in a relation. There is always at least one superkey, because the set of all attributes of the relation is always a superkey (otherwise we could have duplicate tuples).

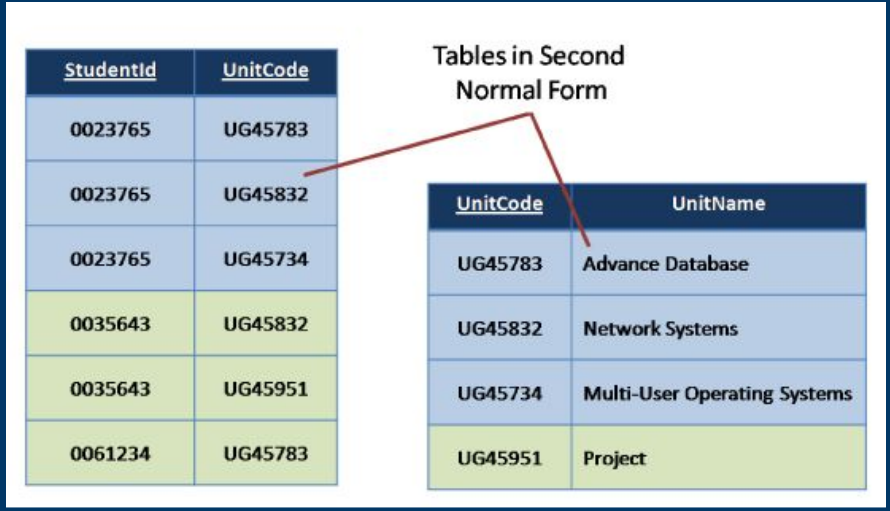
If none of those attributes can be removed and still have it be a superkey, then it's a minimal key, which is also called a candidate key. We choose a single candidate key to be the primary key. If there are any left, they are called alternate keys.

1 Reply Share ...

Second Normal Form (2NF)

1. Must be in First Normal Form
2. Each non-key attribute (column) must depend on the entire Primary Key.
   a. No Partial Key Dependencies

StudentId	UnitCode	UnitName
0023765	UG45783	Advance Database
0023765	UG45832	Network Systems
0023765	UG45734	Multi-User Operating Systems
0035643	UG45832	Network Systems
0035643	UG45951	Project
0061234	UG45783	Advance Database

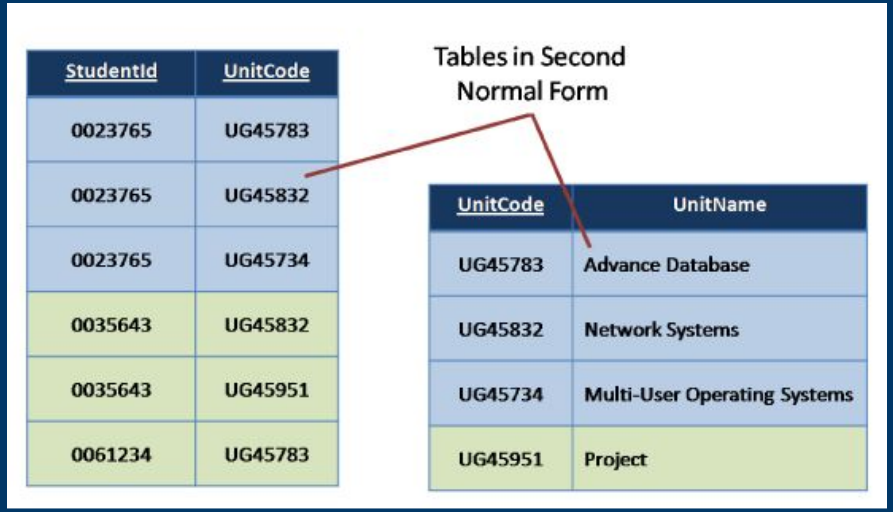


http://rdbms.opengrass.net/2_Database%20Design/2.2_Normalisation/2.2.5_2NF-Partial%20Dependancy.html

Foreign Keys

Foreign Key:

The primary key of one table included in the attributes of a second table to create a link between the two.

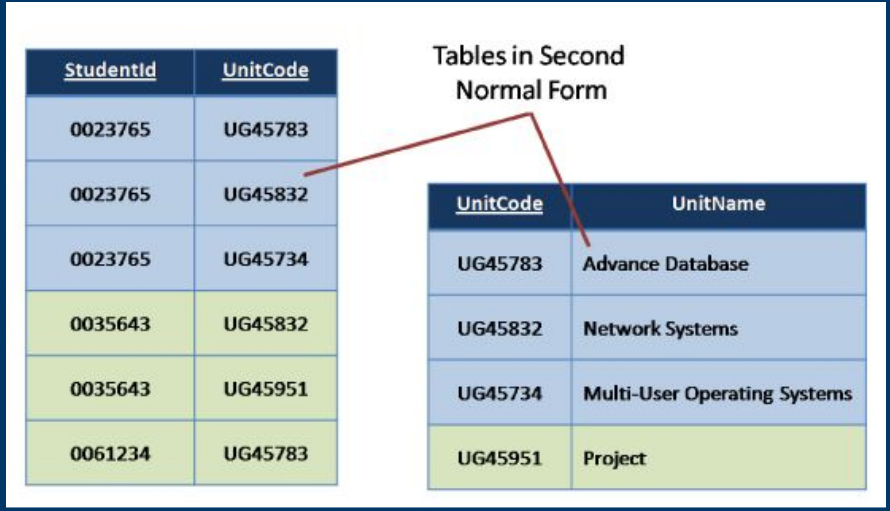


http://rdbms.opengrass.net/2_Database%20Design/2.2_Normalisation/2.2.5_2NF-Partial%20Dependancy.html

Second Normal Form (2NF)

1. Must be in First Normal Form
2. Each non-key attribute (column) must depend on the entire Primary Key.
   a. No Partial Key Dependencies

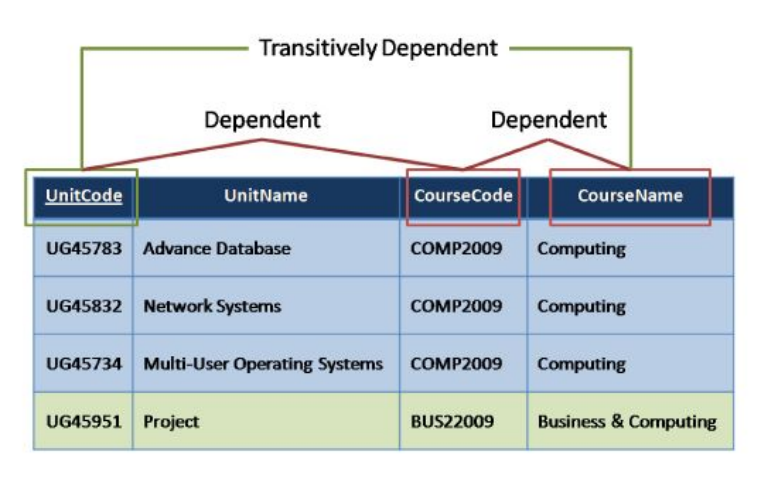
StudentId	UnitCode	UnitName
0023765	UG45783	Advance Database
0023765	UG45832	Network Systems
0023765	UG45734	Multi-User Operating Systems
0035643	UG45832	Network Systems
0035643	UG45951	Project
0061234	UG45783	Advance Database



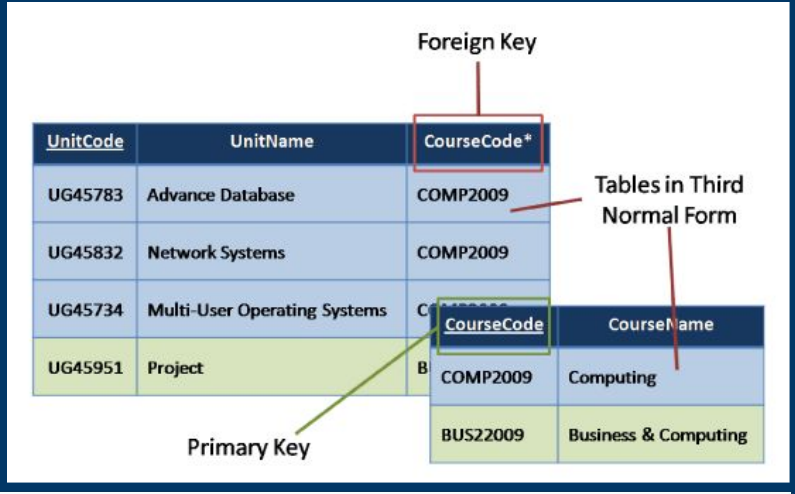
http://rdbms.opengrass.net/2_Database%20Design/2.2_Normalisation/2.2.5_2NF-Partial%20Dependancy.html

Third Normal Form (3NF)

1. Be in 2NF
2. Remove transitive dependencies
   a. If a non-key column depends more on another column than it does the primary key, that's a transitive dependency



UnitCode	UnitName	CourseCode	CourseName
UG45783	Advance Database	COMP2009	Computing
UG45832	Network Systems	COMP2009	Computing
UG45734	Multi-User Operating Systems	COMP2009	Computing
UG45951	Project	BUS22009	Business & Computing



UnitCode	UnitName	CourseCode*
UG45783	Advance Database	COMP2009
UG45832	Network Systems	COMP2009
UG45734	Multi-User Operating Systems	C
UG45951	Project	B

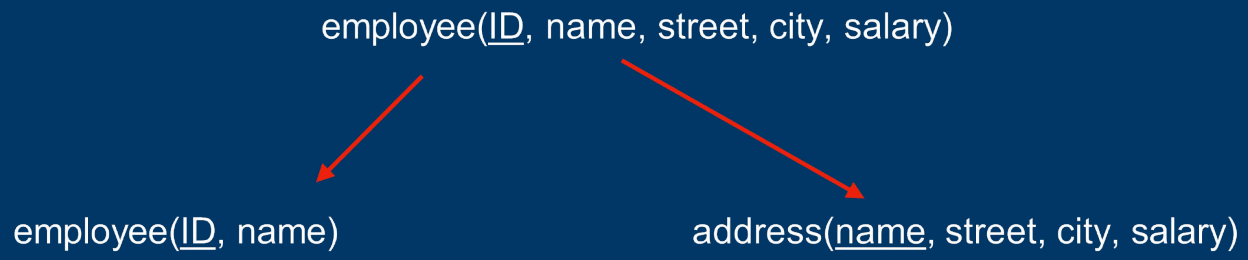
CourseCode	CourseName
COMP2009	Computing
BUS22009	Business & Computing

http://rdbms.opengrass.net/2_Database%20Design/2.2_Normalisation/2.2.6_3NF-Transitive%20Dependency.html

Decomposition Steps

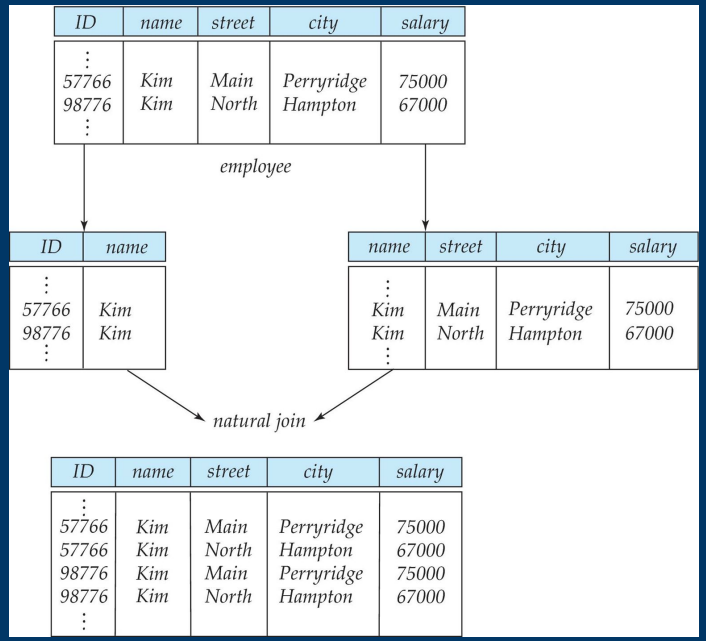
1. Identify Functional Dependencies
2. Apply Normal Forms
3. Ensure Lossless Join
4. Check Dependency Preservation

Lossy Decomposition (Bad)



What if two employees have the same name?
We cannot reconstruct the original *employee* relation from employee and address
this is called a **lossy decomposition**

Lossy



Lossless

ID	name	salary	dept_name	building	budget
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

ID	name	salary	dept_name
22222	Einstein	95000	Physics
12121	Wu	90000	Finance
32343	El Said	60000	History
45565	Katz	75000	Comp. Sci.
98345	Kim	80000	Elec. Eng.
76766	Crick	72000	Biology
10101	Srinivasan	65000	Comp. Sci.
58583	Califieri	62000	History
83821	Brandt	92000	Comp. Sci.
15151	Mozart	40000	Music
33456	Gold	87000	Physics
76543	Singh	80000	Finance

dept_name	building	budget
Physics	Watson	70000
Finance	Painter	120000
History	Painter	50000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Biology	Watson	90000
~~Comp. Sci.~~	~~Taylor~~	~~100000~~
~~History~~	~~Painter~~	~~50000~~
~~Comp. Sci.~~	~~Taylor~~	~~100000~~
Music	Packard	80000
~~Physics~~	~~Watson~~	~~70000~~
~~Finance~~	~~Painter~~	~~120000~~

Functional Dependency

Describes a relationship between two sets of attributes in a database when one attribute uniquely determines another attribute.

If two tuples agree on the values of attributes α they must also agree on the values of attributes β

Questions?

Setting Primary and Foreign Keys

```
CREATE TABLE prereq
      (course_id          VARCHAR(8),
       prereq_id          VARCHAR(8),
       PRIMARY KEY (course_id, prereq_id),
       FOREIGN KEY (course_id) REFERENCES course (course_id)
            ON DELETE CASCADE,
       FOREIGN KEY (prereq_id) REFERENCES course (course_id)
      );
```

ID Numbers as Primary Keys

```
CREATE TABLE classroom
      (ID                 INT AUTO_INCREMENT PRIMARY KEY,
       building           VARCHAR(15),
       room_number        VARCHAR(7),
       capacity           NUMERIC(4,0),
      );
```

UPDATE

We can change the values after our database is created using UPDATE. May need to run “SET SQL_Safe_updates = 0;”

```
UPDATE course
SET CostPerHour = (SELECT DISTINCT(CAST(replace(Cost,"$","") as
decimal(5,2))/ LengthOfTrip) FROM Bookings
WHERE Vessel = "Sea Breeze")
WHERE Name = "Sea Breeze";
```

Table Aliasing

We can give a shorter name to our tables to make referencing them quicker.

```
SELECT *
FROM course c
LEFT JOIN department d
ON c.dept_name = d.dept_name
ORDER BY title DESC;
```

```
SELECT c.course_id, c.title,
       d.dept_name, d.building
FROM course c
LEFT JOIN department d
ON c.dept_name = d.dept_name
ORDER BY title DESC;
```

Clean and Transform

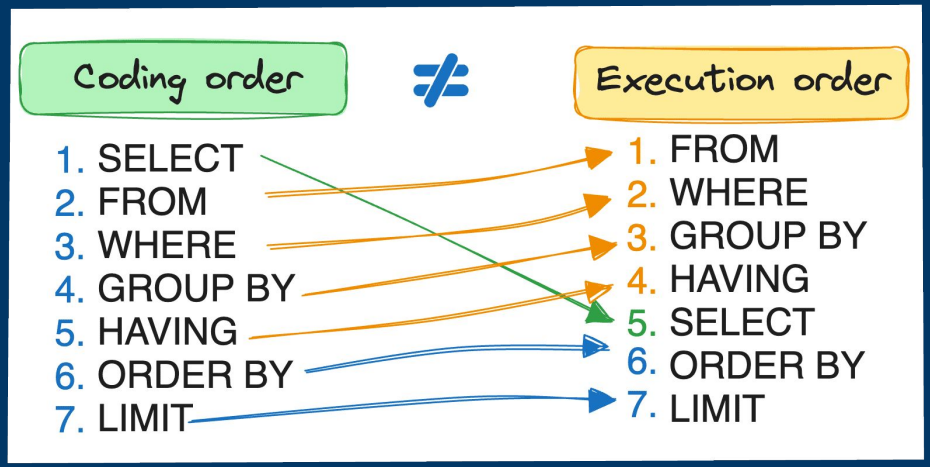
We can also clean and transform our data. For example, we can move values from one table into another.

```
INSERT INTO my_courses
      (my_course_id, my_title)
SELECT takes.course_id, course.title
FROM takes
JOIN course
ON takes.course_id = course.course_id
WHERE student_id = my_student_id;
```

Assumes the existence of a suitable table to insert into.

```
CREATE TABLE my_courses
      (my_course_id    INT,
       my_title VARCHAR(50))...
```

SQL Order of Execution



https://www.kdnuggets.com/the-essential-guide-to-sql-execution-order

MORE ON SELECT

SELECT DISTINCT...
    Removes duplicates from results

SELECT (mathematical or string operation)...
    We can transform our data to get custom results

SELECT (column or expression) AS AnotherName
    Column aliasing lets us rename fields in the results
    The underlying table and data remains the same

CAST

We can display or use values of one datatype as another by using CAST. Using for transforming numbers and dates.

```
SELECT CAST(course_id AS DECIMAL(5,2))
FROM course;

SELECT CAST('123' AS DECIMAL(5,2));
```


https://www.w3schools.com/sql/func_mysql_cast.asp

String Methods

Similar to languages you have used before

CONCAT(s1, v2,...,sn):
    Combine strings and values

REPLACE(text_to_search, text_to_replace, replacement_text):
    Hint: Replace a character with an empty string to delete it

Many more options to review

https://www.w3schools.com/sql/sql_ref_mysql.asp

MORE ON WHERE

>, <, >=, <=, =. <>, LIKE, IS NULL, BETWEEN, IN

Many different comparisons or tests we can use

NOT

We can use negation to further expand our comparison options

AND, OR

We can create complex chains of logic through grouping

ORDERING

You can order by a single column, multiple columns, or a complex expression. The default is ascending (ASC), can also specify descending (DESC)

```
SELECT department_name
FROM department
ORDER BY building_name DESC
```

Nested Subqueries

```
SELECT * FROM course
WHERE dept_name IN (
                        SELECT dept_name
                        FROM department
                        WHERE building LIKE 'T%'
                        )
ORDER BY title DESC;
```

Joins

Match columns from one table with another to combine data.

Natural Join

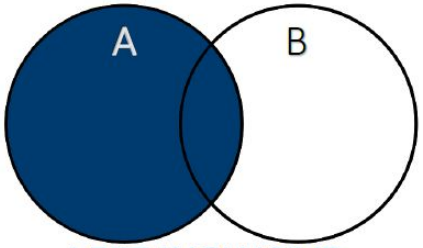
- DBMS decides where to make the relationship

Inner Join

- Excludes rows where they key in one table has no corresponding value

Outer Join (Left, Right, Full)

- Displays null value placeholders when no matching value found



LEFT INCLUSIVE

SQL JOINS

LEFT INCLUSIVE

```
SELECT [Select List]
FROM TableA A
LEFT OUTER JOIN TableB B
ON A.Key= B.Key
```

RIGHT INCLUSIVE

```
SELECT [Select List]
FROM TableA A
RIGHT OUTER JOIN TableB B
ON A.Key= B.Key
```

LEFT EXCLUSIVE

```
SELECT [Select List]
FROM TableA A
LEFT OUTER JOIN TableB B
ON A.Key= B.Key
WHERE B.Key IS NULL
```

RIGHT EXCLUSIVE

```
SELECT [Select List]
FROM TableA A
LEFT OUTER JOIN TableB B
ON A.Key= B.Key
WHERE A.Key IS NULL
```

FULL OUTER INCLUSIVE

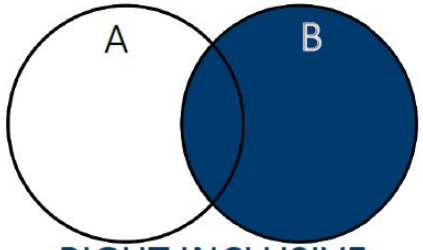
```
SELECT [Select List]
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

FULL OUTER EXCLUSIVE

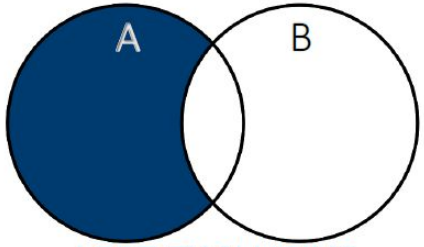
```
SELECT [Select List]
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL OR B.Key IS NULL
```

INNER JOIN

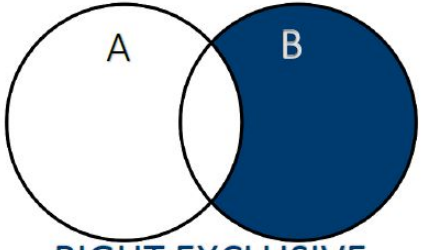
```
SELECT [Select List]
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



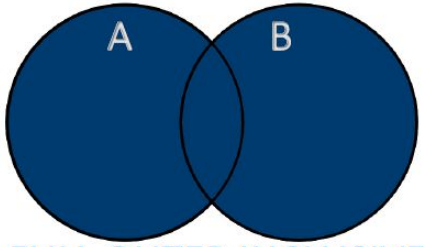
RIGHT INCLUSIVE



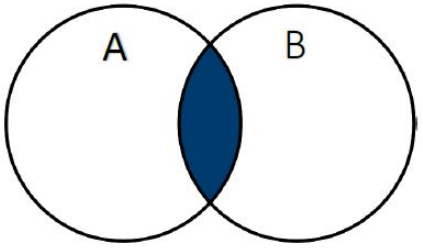
LEFT EXCLUSIVE



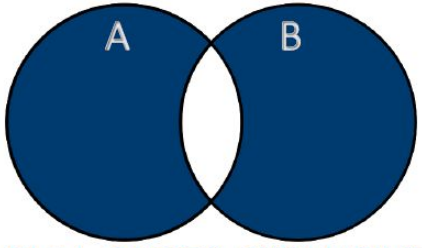
RIGHT EXCLUSIVE



FULL OUTER INCLUSIVE



INNER JOIN



FULL OUTER EXCLUSIVE

https://medium.com/@authfy/seven-join-techniques-in-sql-a65786a40ed3

COURSE SEARCH - Course Search

Academic Year 2024 - 2025 - Spring

Course Search

Course Code

Begins With | csc6

Course Title

Begins With

Division

Instructor

Search...

Department

Search...

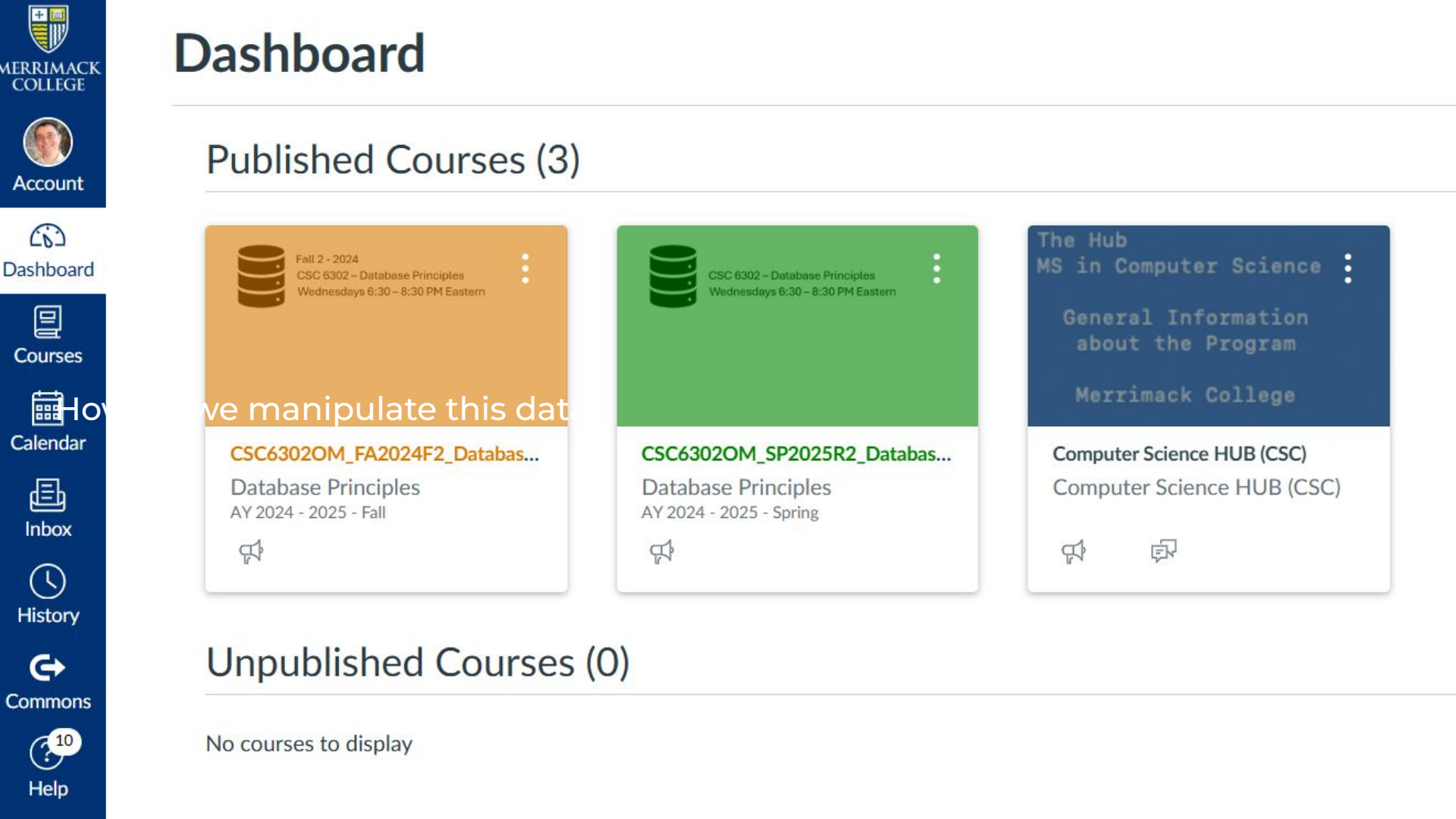
Meeting Type

Meets on Selected Days

M Tu W Th F Sa Su

Search Courses

Course Code	Title	Faculty	Seats Open	Status	Schedule	Credits
CSC-6000-OM	Basic Programming and Discrete M	Paulo Fernandes Paulo Fernandes Paulo Fernandes	65/99	Open	**Thu** 6:30-8:30 PM 1/15/2025 - 3/7/2025 Online Online - OL	4.00
CSC-6000-ON	Basic Programming and Discrete M	Paulo Fernandes Paulo Fernandes Paulo Fernandes	80/99	Open	**Thu** 6:30-8:30 PM 3/17/2025 - 5/9/2025 Online Online - OL	4.00
CSC-6003-OM	Foundations of Programming	Ares Manuel Ca... Ares Manuel Ca... Ares Manuel Ca... Ares Manuel Ca... Ares Manuel Ca...	79/99	Open	**Thu** 6:30-8:30 PM 1/15/2025 - 3/7/2025 Online Online - OL	4.00
CSC-6003-OO	Foundations of Programming	Ares Manuel Ca... Ares Manuel Ca... Ares Manuel Ca... Ares Manuel Ca... Ares Manuel Ca... Sergei Pustylnikov Sergei Pustylnikov Sergei Pustylnikov	68/99	Open	**Thu** 6:30-8:30 PM 3/17/2025 - 5/9/2025 Online Online - OL	4.00
CSC-6013-OM	Algorithms + Discrete Structures	Paulo Fernandes Paulo Fernandes Paulo Fernandes	77/99	Open	**Mon** 6:30-8:30 PM 1/15/2025 - 3/7/2025 Online	4.00


MERRIMACK COLLEGE
Account
Dashboard
Courses
Calendar
Inbox
History
Commons
10
Help
Dashboard
Published Courses (3)
Fall 2 - 2024
CSC 6302 – Database Principles
Wednesdays 6:30 – 8:30 PM Eastern
CSC6302OM_FA2024F2_Databas...
Database Principles
AY 2024 - 2025 - Fall
CSC 6302 – Database Principles
Wednesdays 6:30 – 8:30 PM Eastern
CSC6302OM_SP2025R2_Databas...
Database Principles
AY 2024 - 2025 - Spring
The Hub
MS in Computer Science
General Information about the Program
Merrimack College
Computer Science HUB (CSC)
Computer Science HUB (CSC)
Unpublished Courses (0)
No courses to display

Questions?

Project 2:

After setting up the MRC database, you realize it would be much better if the you split up the bookings table into three separate tables that you can join together later. This also gives you the opportunity to alter some of the datatypes to be more useful for aggregation.

Opening Starter Code

Easiest way: be in your local instance tab and then open the file using the open file dialog.