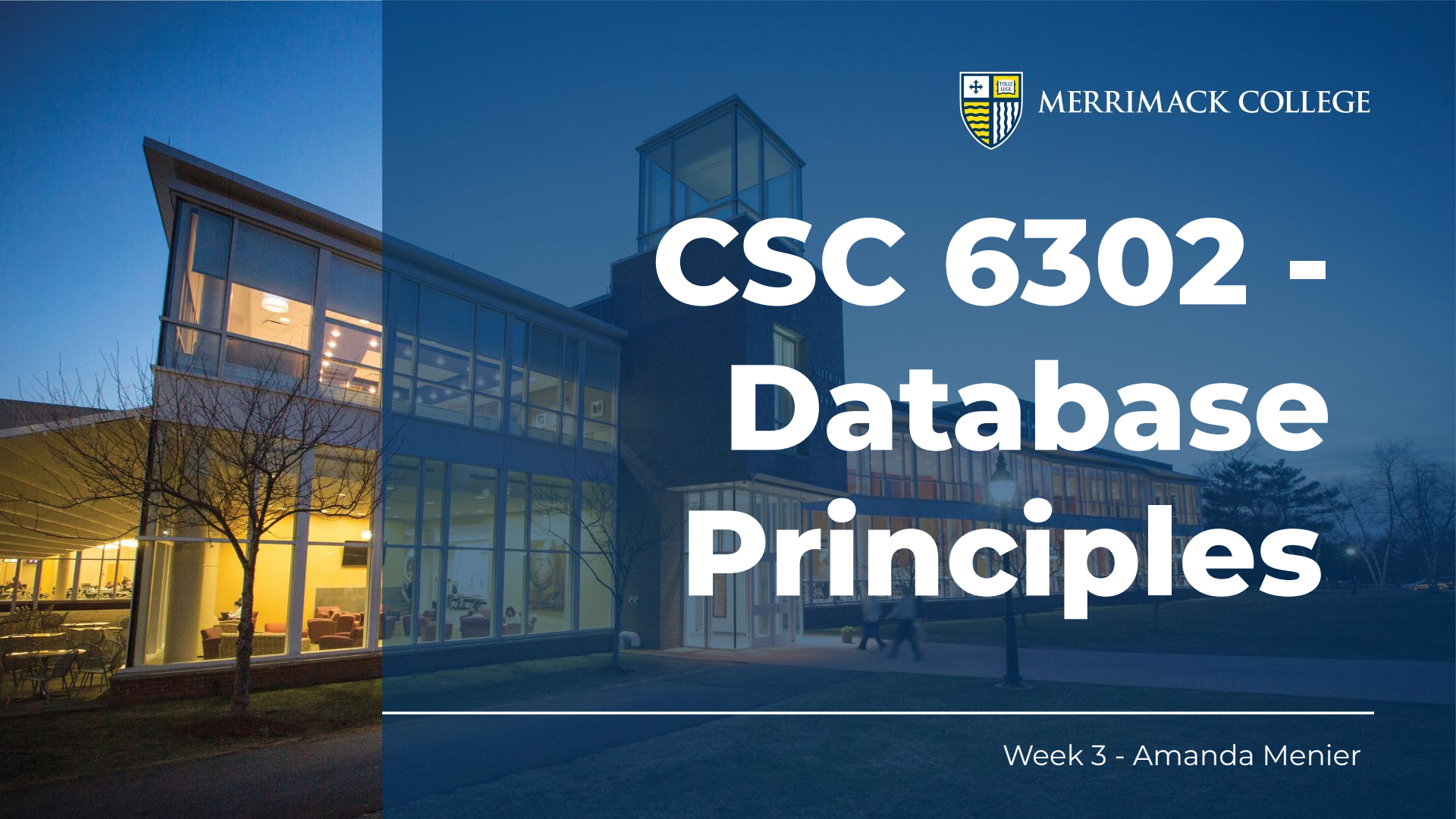


MERRIMACK COLLEGE

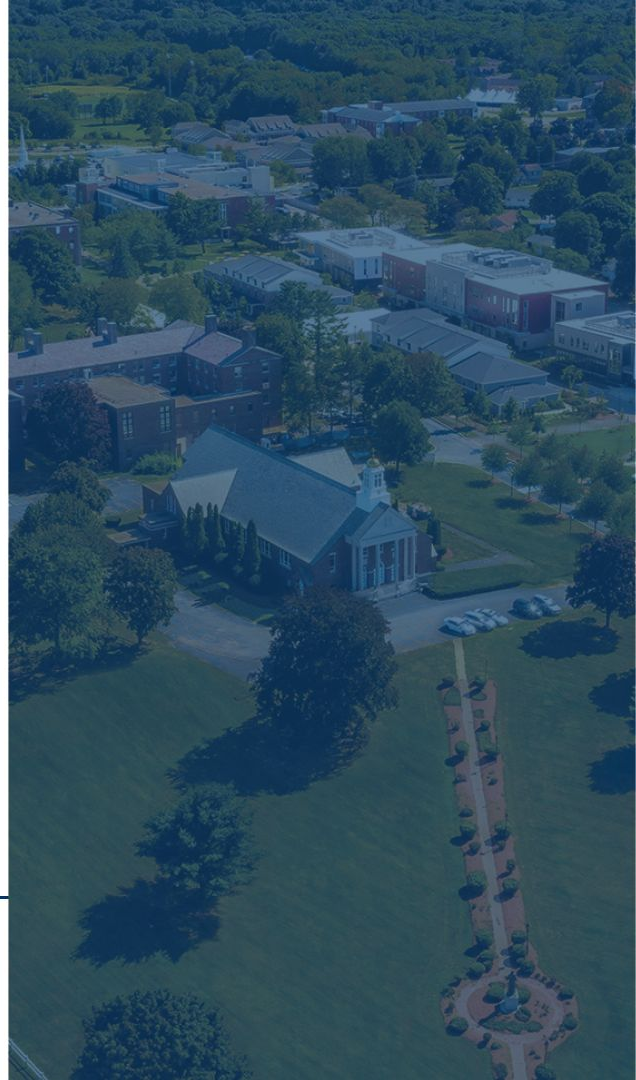
CSC 6302 - Database Principles

Week 3 - Amanda Menier

Agenda

Discussion Topics:

- Week 2 Review
- ER Diagrams & Cardinality
- Aggregation
- Transactions & Delimiters
- Views
- Functions
- Procedures



Project 2 Notes

1) There are 12 superkeys. The candidate keys are {Date, Departure_Time, Vessel_ID} and {Date, Departure_Time, Passenger_ID}. 2^n - 1 is the number of potential superkeys, but that's only true if they would all uniquely identify a row.
2) Starter code file should be used as-is - put your code below.
3) Cost per hours should be associated with a vessel, not a trip.
4) Watch out for details such as the number of decimal places.

All Potential Superkeys

{Vessel_ID},
{Passenger_ID},
{Date},
{Departure_Time},
{Length_in_Hours},
{Total_Passengers},
{Vessel_ID, Passenger_ID},
{Vessel_ID, Date},
{Passenger_ID, Date},
{Vessel_ID, Departure_Time},
{Passenger_ID, Departure_Time},
{Date, Departure_Time},
{Vessel_ID, Length_in_Hours},
{Passenger_ID, Length_in_Hours},
{Date, Length_in_Hours},
{Departure_Time, Length_in_Hours},
{Vessel_ID, Total_Passengers},
{Passenger_ID, Total_Passengers},
{Date, Total_Passengers},
{Departure_Time, Total_Passengers},
{Length_in_Hours, Total_Passengers},
{Vessel_ID, Passenger_ID, Date},
{Vessel_ID, Passenger_ID, Departure_Time},
{Vessel_ID, Date, Departure_Time},
{Passenger_ID, Date, Departure_Time},
{Vessel_ID, Passenger_ID, Length_in_Hours},
{Vessel_ID, Date, Length_in_Hours},
{Passenger_ID, Date, Length_in_Hours},
{Vessel_ID, Departure_Time, Length_in_Hours},
{Passenger_ID, Departure_Time, Length_in_Hours},
{Date, Departure_Time, Length_in_Hours},
{Vessel_ID, Passenger_ID, Total_Passengers},
{Vessel_ID, Date, Total_Passengers},
{Passenger_ID, Date, Total_Passengers},

{Vessel_ID, Departure_Time, Total_Passengers},
{Passenger_ID, Departure_Time, Total_Passengers},
{Date, Departure_Time, Total_Passengers},
{Vessel_ID, Length_in_Hours, Total_Passengers},
{Passenger_ID, Length_in_Hours, Total_Passengers},
{Date, Length_in_Hours, Total_Passengers},
{Departure_Time, Length_in_Hours, Total_Passengers},
{Vessel_ID, Passenger_ID, Date, Departure_Time},
{Vessel_ID, Passenger_ID, Date, Length_in_Hours},
{Vessel_ID, Passenger_ID, Departure_Time, Length_in_Hours},
{Vessel_ID, Date, Departure_Time, Length_in_Hours},
{Passenger_ID, Date, Departure_Time, Length_in_Hours},
{Vessel_ID, Passenger_ID, Date, Total_Passengers},
{Vessel_ID, Passenger_ID, Departure_Time, Total_Passengers},
{Vessel_ID, Date, Departure_Time, Total_Passengers},
{Passenger_ID, Date, Departure_Time, Total_Passengers},
{Vessel_ID, Passenger_ID, Length_in_Hours, Total_Passengers},
{Vessel_ID, Date, Length_in_Hours, Total_Passengers},
{Passenger_ID, Date, Length_in_Hours, Total_Passengers},
{Vessel_ID, Departure_Time, Length_in_Hours, Total_Passengers},
{Passenger_ID, Departure_Time, Length_in_Hours, Total_Passengers},
{Date, Departure_Time, Length_in_Hours, Total_Passengers},
{Vessel_ID, Passenger_ID, Date, Departure_Time, Length_in_Hours},
{Vessel_ID, Passenger_ID, Date, Departure_Time, Total_Passengers},
{Vessel_ID, Passenger_ID, Date, Length_in_Hours, Total_Passengers},
{Vessel_ID, Passenger_ID, Departure_Time, Length_in_Hours, Total_Passengers},
{Vessel_ID, Date, Departure_Time, Length_in_Hours, Total_Passengers},
{Passenger_ID, Date, Departure_Time, Length_in_Hours, Total_Passengers},
{Vessel_ID, Passenger_ID, Date, Departure_Time, Length_in_Hours, Total_Passengers}

{Vessel_ID},
{Passenger_ID},
{Date},
{Departure_Time},
{Length_in_Hours},
{Total_Passengers},
{Vessel_ID, Passenger_ID},
{Vessel_ID, Date},
{Passenger_ID, Date},
{Vessel_ID, Departure_Time},
{Passenger_ID, Departure_Time},
{Date, Departure_Time},
{Vessel_ID, Length_in_Hours},
{Passenger_ID, Length_in_Hours},
{Date, Length_in_Hours},
{Departure_Time, Length_in_Hours},
{Vessel_ID, Total_Passengers},
{Passenger_ID, Total_Passengers},
{Date, Total_Passengers},
{Departure_Time, Total_Passengers},
{Length_in_Hours, Total_Passengers},
{Vessel_ID, Passenger_ID, Date},
{Vessel_ID, Passenger_ID, Departure_Time},
{Vessel_ID, Date, Departure_Time},
{Passenger_ID, Date, Departure_Time},
{Vessel_ID, Passenger_ID, Length_in_Hours},
{Vessel_ID, Date, Length_in_Hours},
{Passenger_ID, Date, Length_in_Hours},
{Vessel_ID, Departure_Time, Length_in_Hours},
{Passenger_ID, Departure_Time, Length_in_Hours},
{Date, Departure_Time, Length_in_Hours},
{Vessel_ID, Passenger_ID, Total_Passengers},
{Vessel_ID, Date, Total_Passengers},
{Passenger_ID, Date, Total_Passengers},

12 Superkeys

{Vessel_ID, Departure_Time, Total_Passengers},
{Passenger_ID, Departure_Time, Total_Passengers},
{Date, Departure_Time, Total_Passengers},
{Vessel_ID, Length_in_Hours, Total_Passengers},
{Passenger_ID, Length_in_Hours, Total_Passengers},
{Date, Length_in_Hours, Total_Passengers},
{Departure_Time, Length_in_Hours, Total_Passengers},
{Vessel_ID, Passenger_ID, Date, Departure_Time},
{Vessel_ID, Passenger_ID, Date, Length_in_Hours},
{Vessel_ID, Passenger_ID, Departure_Time, Length_in_Hours},
{Vessel_ID, Date, Departure_Time, Length_in_Hours},
{Passenger_ID, Date, Departure_Time, Length_in_Hours},
{Vessel_ID, Passenger_ID, Date, Total_Passengers},
{Vessel_ID, Passenger_ID, Departure_Time, Total_Passengers},
{Vessel_ID, Date, Departure_Time, Total_Passengers},
{Passenger_ID, Date, Departure_Time, Total_Passengers},
{Vessel_ID, Passenger_ID, Length_in_Hours, Total_Passengers},
{Vessel_ID, Date, Length_in_Hours, Total_Passengers},
{Passenger_ID, Date, Length_in_Hours, Total_Passengers},
{Vessel_ID, Departure_Time, Length_in_Hours, Total_Passengers},
{Passenger_ID, Departure_Time, Length_in_Hours, Total_Passengers},
{Date, Departure_Time, Length_in_Hours, Total_Passengers},
{Vessel_ID, Passenger_ID, Date, Departure_Time, Length_in_Hours},
{Vessel_ID, Passenger_ID, Date, Departure_Time, Total_Passengers},
{Vessel_ID, Passenger_ID, Date, Length_in_Hours, Total_Passengers},
{Vessel_ID, Passenger_ID, Departure_Time, Length_in_Hours, Total_Passengers},
{Vessel_ID, Date, Departure_Time, Length_in_Hours, Total_Passengers},
{Passenger_ID, Date, Departure_Time, Length_in_Hours, Total_Passengers},
{Vessel_ID, Passenger_ID, Date, Departure_Time, Length_in_Hours, Total_Passengers}

COLLEGE

(Vessel_ID),
(Passenger_ID),
(Date),
(Departure_Time),
(Length_in_Hours),
(Total_Passengers),
(Vessel_ID, Passenger_ID),
(Vessel_ID, Date),
(Passenger_ID, Date),
(Vessel_ID, Departure_Time),
(Passenger_ID, Departure_Time),
(Date, Departure_Time),
(Vessel_ID, Length_in_Hours),
(Passenger_ID, Length_in_Hours),
(Date, Length_in_Hours),
(Departure_Time, Length_in_Hours),
(Vessel_ID, Total_Passengers),
(Passenger_ID, Total_Passengers),
(Date, Total_Passengers),
(Departure_Time, Total_Passengers),
(Length_in_Hours, Total_Passengers),
(Vessel_ID, Passenger_ID, Date),
(Vessel_ID, Passenger_ID, Departure_Time),
{Vessel_ID, Date, Departure_Time},
{Passenger_ID, Date, Departure_Time},
(Vessel_ID, Passenger_ID, Length_in_Hours),
(Vessel_ID, Date, Length_in_Hours),
(Passenger_ID, Date, Length_in_Hours),
(Vessel_ID, Departure_Time, Length_in_Hours),
(Passenger_ID, Departure_Time, Length_in_Hours),
(Date, Departure_Time, Length_in_Hours),
(Vessel_ID, Passenger_ID, Total_Passengers),
(Vessel_ID, Date, Total_Passengers),
(Passenger_ID, Date, Total_Passengers),

(Vessel_ID, Departure_Time, Total_Passengers),
(Passenger_ID, Departure_Time, Total_Passengers),
(Date, Departure_Time, Total_Passengers),
(Vessel_ID, Length_in_Hours, Total_Passengers),
(Passenger_ID, Length_in_Hours, Total_Passengers),
(Date, Length_in_Hours, Total_Passengers),
(Departure_Time, Length_in_Hours, Total_Passengers),
(Vessel_ID, Passenger_ID, Date, Departure_Time),
(Vessel_ID, Passenger_ID, Date, Length_in_Hours),
(Vessel_ID, Passenger_ID, Departure_Time, Length_in_Hours),
(Vessel_ID, Date, Departure_Time, Length_in_Hours),
(Passenger_ID, Date, Departure_Time, Length_in_Hours),
(Vessel_ID, Passenger_ID, Date, Total_Passengers),
(Vessel_ID, Passenger_ID, Departure_Time, Total_Passengers),
(Vessel_ID, Date, Departure_Time, Total_Passengers),
(Passenger_ID, Date, Departure_Time, Total_Passengers),
(Vessel_ID, Passenger_ID, Length_in_Hours, Total_Passengers),
(Vessel_ID, Date, Length_in_Hours, Total_Passengers),
(Passenger_ID, Date, Length_in_Hours, Total_Passengers),
(Vessel_ID, Departure_Time, Length_in_Hours, Total_Passengers),
(Passenger_ID, Departure_Time, Length_in_Hours, Total_Passengers),
(Date, Departure_Time, Length_in_Hours, Total_Passengers),
(Vessel_ID, Passenger_ID, Date, Departure_Time, Length_in_Hours),
(Vessel_ID, Passenger_ID, Date, Departure_Time, Total_Passengers),
(Vessel_ID, Passenger_ID, Date, Length_in_Hours, Total_Passengers),
(Vessel_ID, Passenger_ID, Departure_Time, Length_in_Hours, Total_Passengers),
(Vessel_ID, Date, Departure_Time, Length_in_Hours, Total_Passengers),
(Passenger_ID, Date, Departure_Time, Length_in_Hours, Total_Passengers),
(Vessel_ID, Passenger_ID, Date, Departure_Time, Length_in_Hours, Total_Passengers),

2 Candidate Keys

COLLEGE

Keys

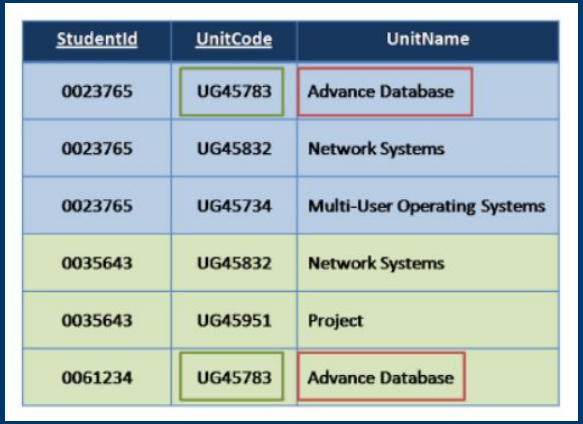
Decomposition: Breaking complex tables into simpler ones that preserve information

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

Candidate Key: A superkey that uses the fewest attributes necessary

Primary Key: A candidate key we choose to identify tuples in our relation based on our Needs and goals

Normal Forms



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

First Normal Form (1NF):
- Row order doesn't store information
- Single datatype in each column
- Must have primary key
- No repeating data groups
- Unique names for columns

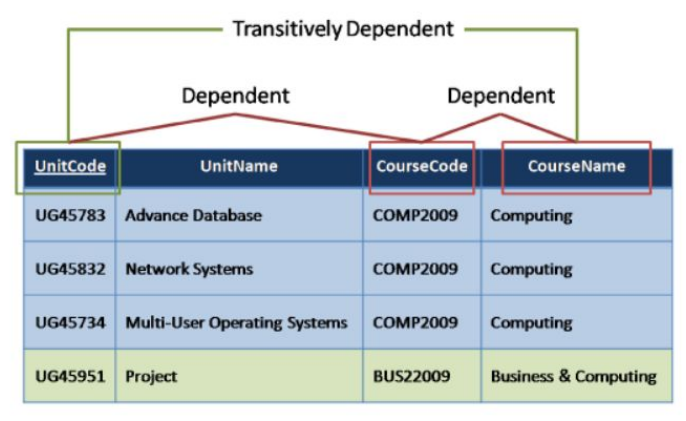
Second Normal Form (2NF):
- Be in 1NF
- No partial key dependencies

Third Normal Form (3NF):
- Be in 2NF
- No transitive dependencies

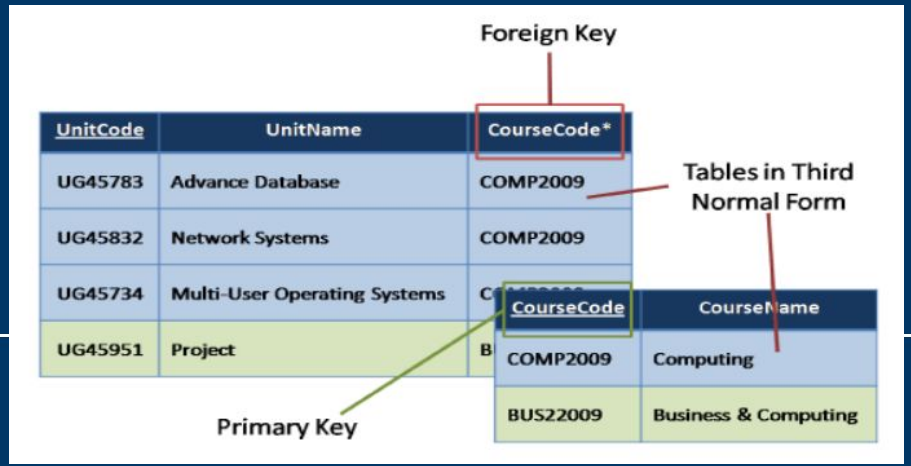
Transitive Dependencies

Functional Dependency: For K → R:

In other words, if two tuples agree on the values of the attributes in K, they must also agree on the values of all attributes in R.



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

Joins

We can match up primary and foreign keys in different tables to display more information.

Natural Join - Join on all matching attributes - lets DBMS decide which attributes

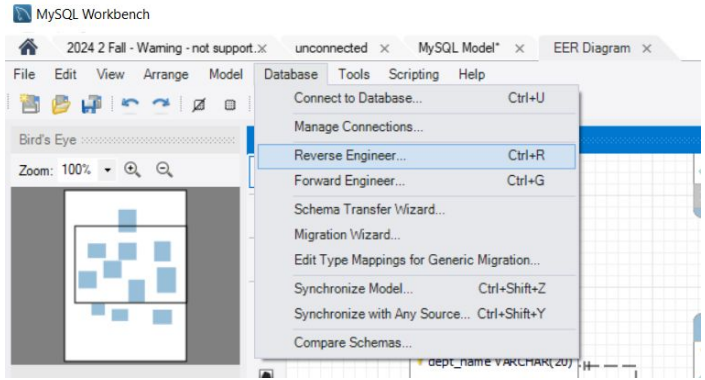
Inner Join - Joining attribute must be present in both both tables to join tuples - excludes those with null values or that don't have a match

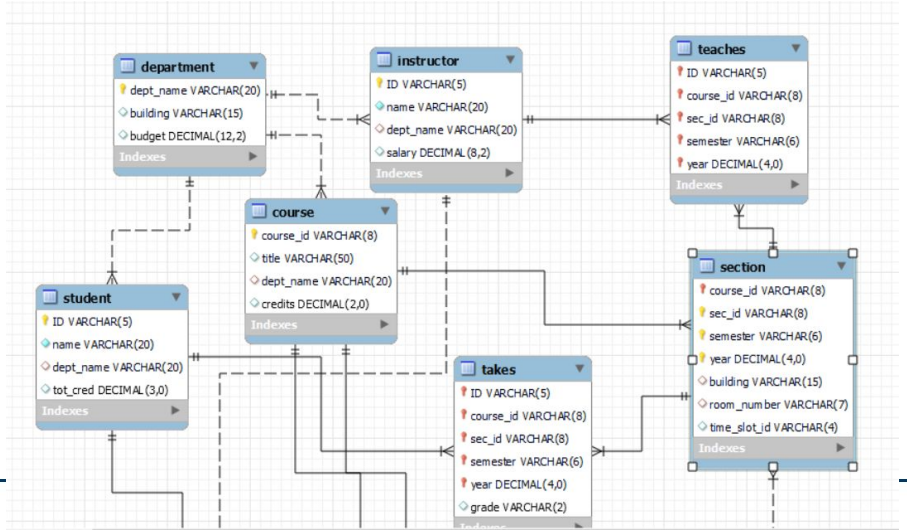
Left Outer Join - All tuples from the left table with matching tuples from the right, or null values

Right Outer Join - All tuples from the right table with matching tuples from the left, or null values

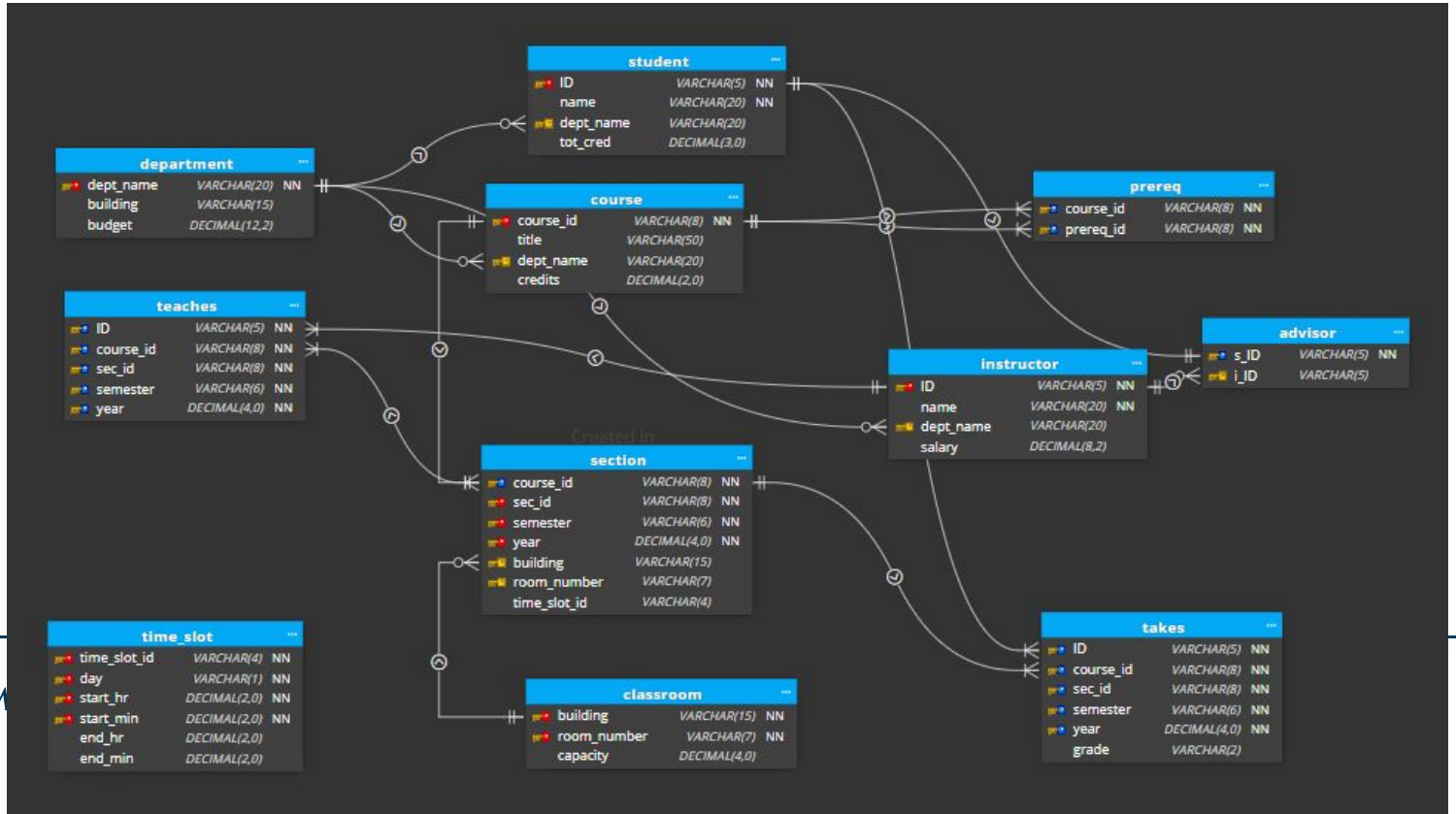
Entity Relationship Diagrams

The great advantage of using a relational database is creating relationships between our tables. This gives us efficiencies in storing and retrieving information. We can map these relationships using an Entity Relationship Diagram (ERD).





Luna Modeler

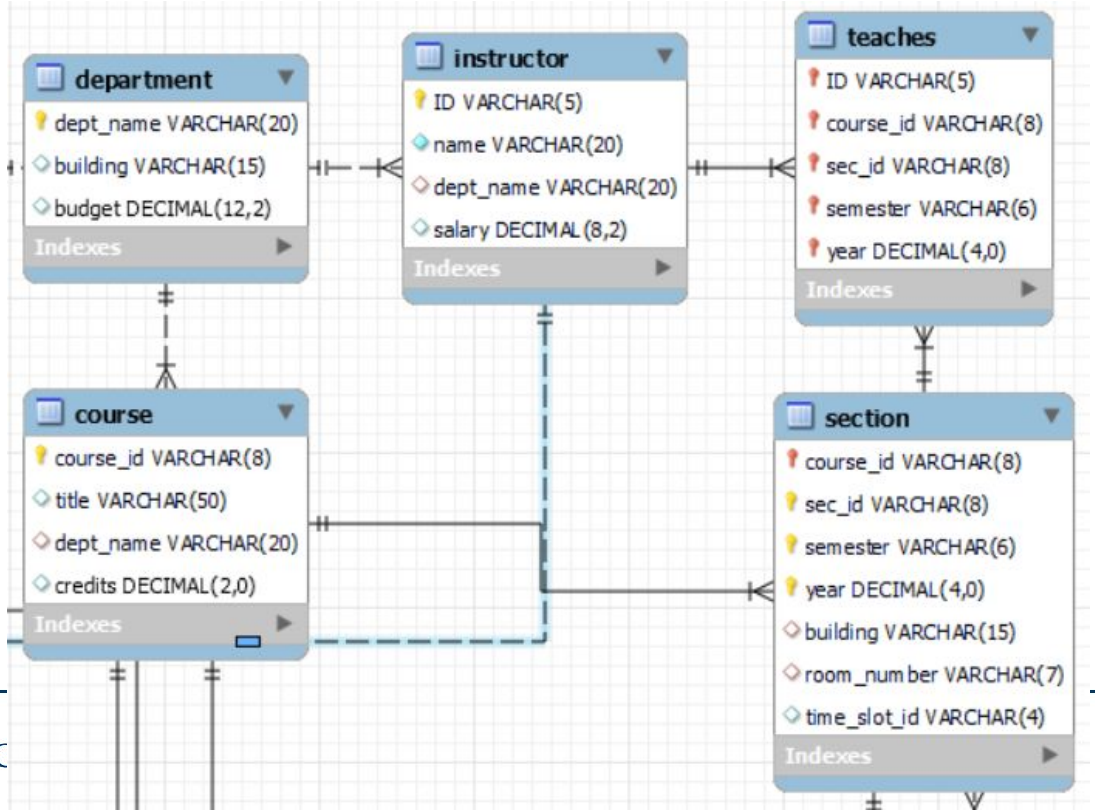


Cardinality

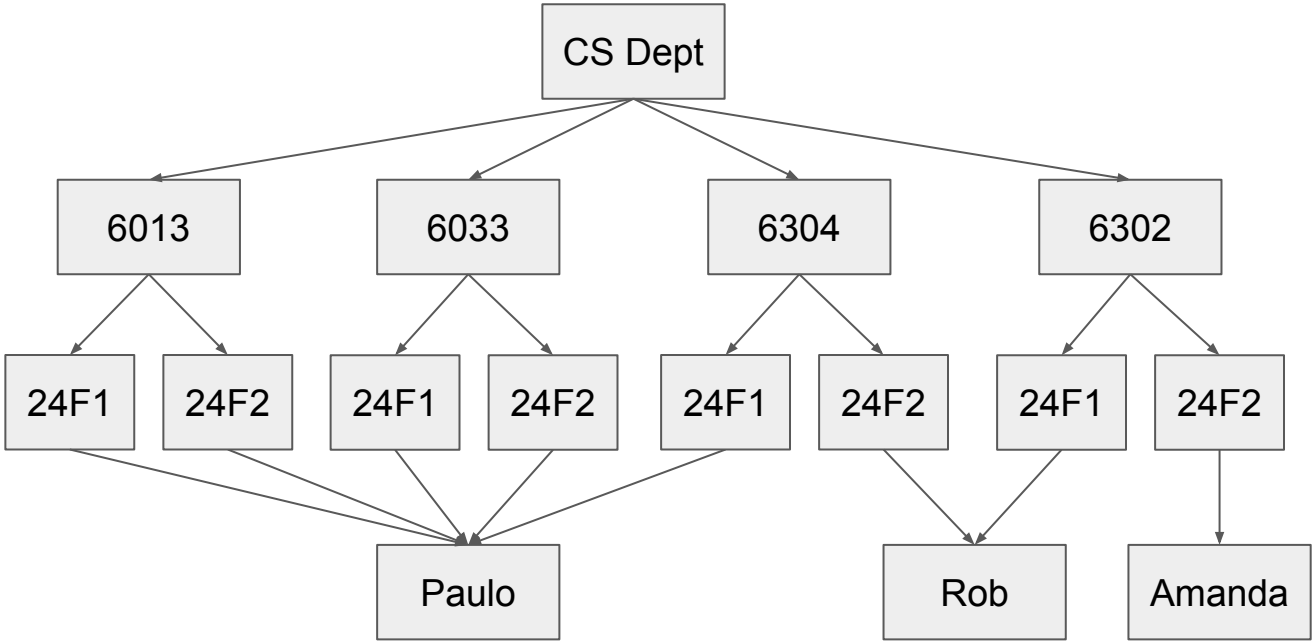
When we create relationships between tables, we need to specify cardinality. Often the RDBMS will infer this for us, but we may need to take manual control. Understanding cardinality will help us predict and troubleshoot the results when we join our tables. There are three cardinalities we should be concerned about.

- One to One - The first table and second table only have one tuple each that will be matched together
- One to Many - The first table has unique values in the matched attribute, and the second table has repeated values in the matched attribute
- Many to Many - The first table has repeated values in the matched attribute and the second table **also** has repeated values in the matched attribute

Cardinality Example



Seem Familiar?



Joined Table

Pseudocode:

SELECT course information

FROM Department

JOIN courses ON dept = dept

JOIN sections ON course# = course#

JOIN instructor ON name = name

WHERE dept = ‘CS Dept’

AND term = ‘Fall 24’

dept	course#	section	name
CS Dept	6013	24F1	Paulo
CS Dept	6013	24F2	Paulo
CS Dept	6033	24F1	Paulo
CS Dept	6033	24F2	Paulo
CS Dept	6304	24F1	Paulo
CS Dept	6304	24F2	Rob
CS Dept	6302	24F1	Rob
CS Dept	6302	24F2	Amanda

Aggregation

We can have SQL sum, average, min, max, and more!
To do so, we use the aggregation function in our SELECT statement.
We then need to include all non-aggregated columns in the GROUP BY statement.
We can filter by aggregated fields by using a HAVING statement

```
SELECT dept_name, sum(salary)
FROM instructor
GROUP BY dept_name
HAVING sum(salary) > 100000;
```

Views

Views are virtual relations. They allow us to repeat a saved query in the future. If we know we will be running a query often, we can define a view.

Advantages of views:

- We can query them just like any other relation
- We can even make views from views if we want

Basic syntax:

```
CREATE VIEW viewname AS
    SELECT x
    FROM y
```

View Examples

A view of instructors without their salary

```
create view faculty as
        select ID, name, dept_name
        from instructor
```

Find all instructors in the Biology

```
select name
from faculty
where dept_name = 'Biology'
```

Create a view of department salary totals

```
create view departments_total_salary(dept_name, total_salary) as
        select dept_name, sum(salary)
        from instructor
        group by dept_name;
```

Transactions

Transactions group a series of statements, queries, and actions together. This happens implicitly when we execute a statement. This means

- Each transaction is a unit of work (atomic transactions)
- Transaction must end in a commit or a rollback
  a. Commit: Updates become permanent
  b. Rollback: All updates in the transaction are undone
- If the transaction does not complete, it's as if it never happened
- This separates transactions that happen at the same time

Delimiters

Delimiters tell the code when a statement ends. MySQL uses a semicolon as default. When we want to group a set of statements that end in a semicolon together, we can temporarily change the delimiter. We will need to do this to create functions and stored procedures. It will look something like:

```
delimiter $$
	BEGIN
		STATEMENT A;
		STATEMENT B;
	END$$

delimiter ;
```

Variables

As in other languages, we can specify variables. They can be included in a script, used in a block of code for a function or procedure, or set system level variables.

Session Scope - Used in a SQL Script:

```
set @start=1;
set @end=10;
select @start;
select @end;
select * from Series where Id between @start and @end;
select @middle := 5;
select * from Series where Id between @start and @middle;
```

Variables

Local Variables:
Used in the scope of a single function or stored procedure
Defined with a begin/end block

System Variables:
Used only for system admin tasks
Can have global or session scope
Prefaced with @@

Functions

Functions are repeatable snippets of code that return a value or values. They have different syntax, but are conceptually the same as functions we see in other languages. A common use case is to get the id of some item or person:

```
DROP FUNCTION IF EXISTS GetFacultyId;
delimiter $$
CREATE FUNCTION GetFacultyId (desiredFacultyName varchar(200)) RETURNS INT
deterministic
BEGIN
        declare desiredId int;
        SELECT distinct ID into desiredId
        FROM instructor
        WHERE name = desiredFacultyName;
  return desiredId;
END$$
delimiter ;
```

Functions can be used in queries and procedures. Example:
SELECT GetFacultyID('Einstein');

Procedures

Procedures are also repeatable groups of statements that can use queries, variables, views, functions, and so forth. They are valuable because they allow us to define common actions and call them later. The applications we will build later this term will use procedures to “safely” run predefined queries and operations.

```
DROP PROCEDURE IF EXISTS GetAllCourses;
delimiter $$
   CREATE PROCEDURE GetAllCourses (desiredDeptName varchar(200))
   BEGIN
      SELECT *
      FROM course
      WHERE dept_name = desiredDeptName;
   END;
   $$
delimiter ;
```

Procedures are ‘called.’
Example:
CALL GetAllCourses(‘Comp.Sci.’);

More Procedures

```
create procedure dept_count_proc (in dept_name varchar(20), out d_count integer)
begin
        select count(*) into d_count
        from instructor
        where instructor.dept_name = dept_count_proc.dept_name
end
```

- The keywords in and out are parameters that are expected to have values assigned to them and parameters whose values are set in the procedure in order to return results.
- Procedures can be invoked either from an SQL procedure or from embedded SQL, using the call statement.

```
declare d_count integer;
call dept_count_proc( 'Physics', d_count)
```

Project 3

We are going to use functions, views, and procedures to add value to our booking system.