Introduction to Management Information Systems

Relational Database Design

Data Resource Management

Relational Database: Junction Table

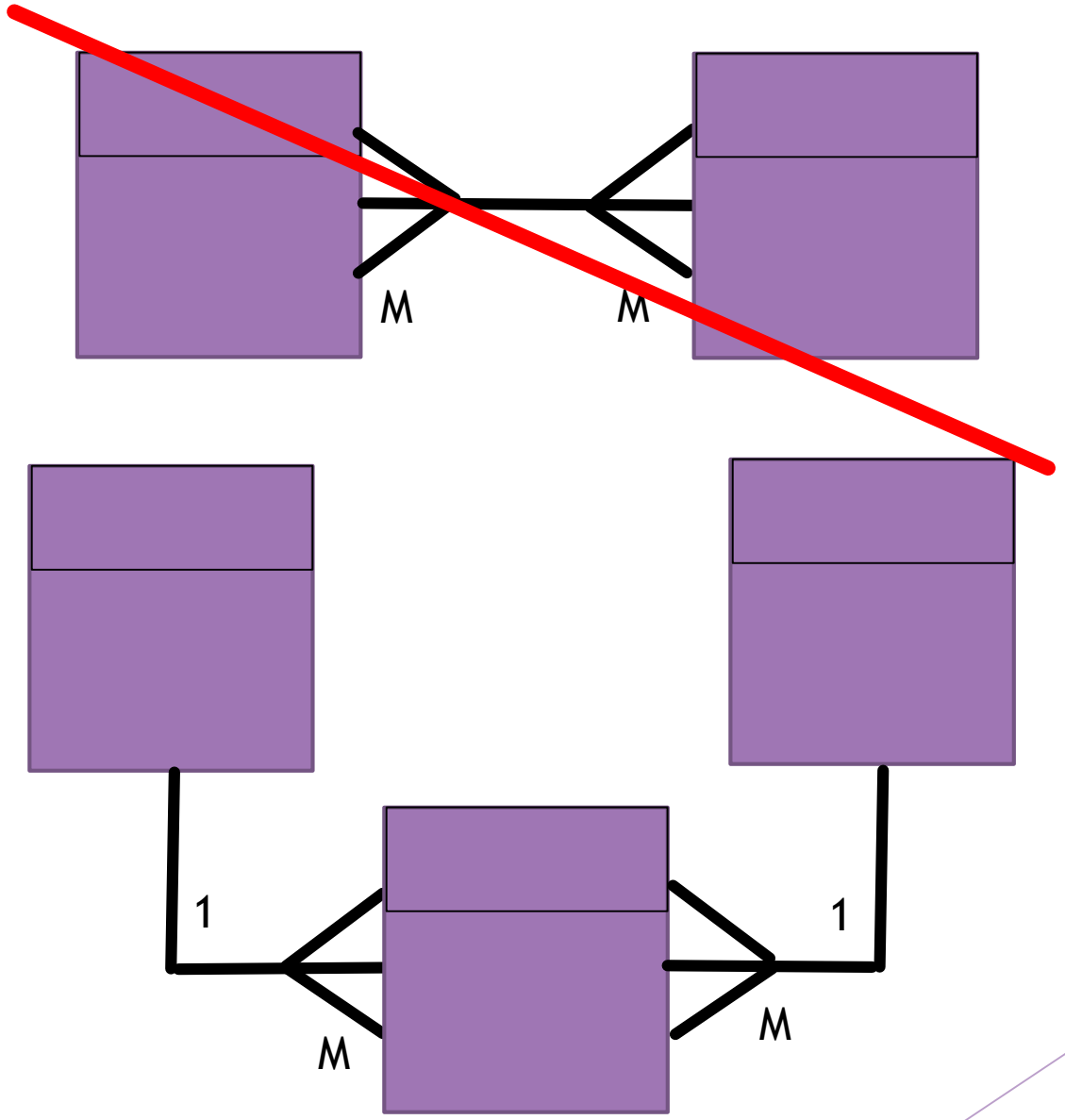
implement our design

1. Using a Junction Table
2. SQL for Multiple Tables
3. Database Design considerations

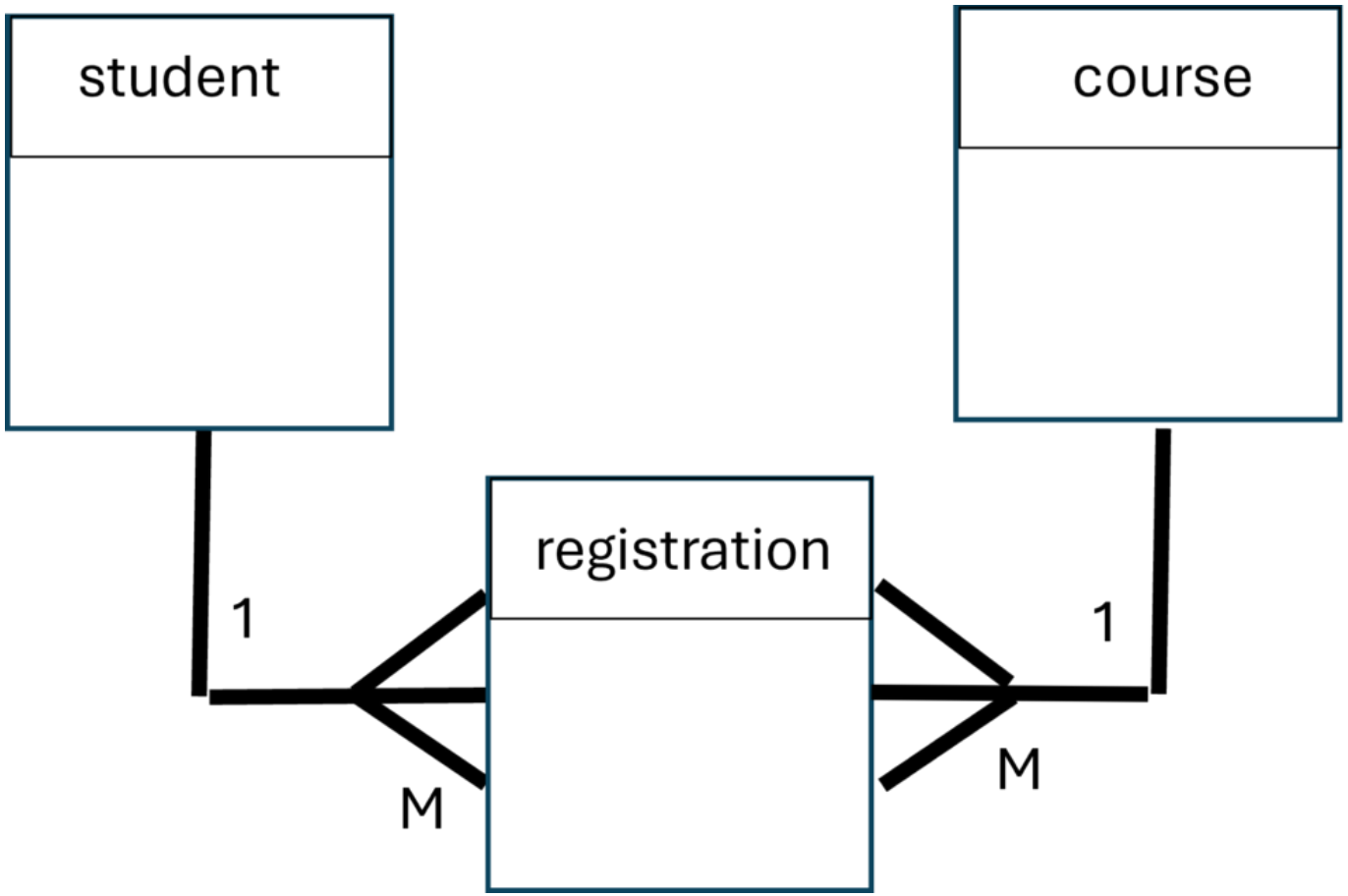
our task today is to

- make our database relational
  - see multiple table data

Junction Table



Junction Table



student table

```
CREATE TABLE student (
student_id INT PRIMARY KEY,
student_name VARCHAR(100)
);
```

primary key in another table is a foreign key

```
FOREIGN KEY (student_id) REFERENCES student(student_id)
```

```
CREATE TABLE student (
student_id INT PRIMARY KEY,
student_name VARCHAR(100)
);
```

primary key in another table is a foreign key

FOREIGN KEY (student_id) REFERENCES student(student_id)

```
CREATE TABLE registration (
    reg_id INT PRIMARY KEY,
    student_id INT,
    course_code INT,
    FOREIGN KEY (student_id) REFERENCES student(student_id),
    FOREIGN KEY (course_code) REFERENCES course(course_code)
);
```

primary key in another table is a foreign key

FOREIGN KEY (student_id) REFERENCES student(student_id)

```
CREATE TABLE registration (
    reg_id INT PRIMARY KEY,
    student_id INT,
    course_code INT,
    FOREIGN KEY (student_id) REFERENCES student(student_id),
    FOREIGN KEY (course_code) REFERENCES course(course_code)
);
```

primary key in another table is a foreign key

FOREIGN KEY (course_code) REFERENCES course(course_code)

```
CREATE TABLE registration (
    reg_id INT PRIMARY KEY,
    student_id INT,
    course_code INT,
    FOREIGN KEY (student_id) REFERENCES student(student_id),
    FOREIGN KEY (course_code) REFERENCES course(course_code)
);
```

create registration table

```
CREATE TABLE registration (
    reg_id INT PRIMARY KEY,
    student_id INT,
    course_code INT,
    FOREIGN KEY (student_id) REFERENCES student(student_id),
    FOREIGN KEY (course_code) REFERENCES course(course_code)
);
```

insert data into the registration table

```
INSERT INTO registration (reg_id, student_id, course_code)
VALUES
(3001, 652415501, 888342),
(3002, 652415502, 888342),
(3003, 652415503, 881234),
(3004, 652415503, 888342),
(3005, 652415504, 888341),
(3006, 652415504, 888342),
(3007, 652415505, 883123),
(3008, 652415505, 881234),
(3009, 652415506, 884567),
(3010, 652415506, 888341),
(3011, 652415507, 885678),
(3012, 652415508, 886345),
(3013, 652415508, 888342),
(3014, 652415509, 887654),
(3015, 652415509, 888341),
(3016, 652415510, 888120),
(3017, 652415510, 884567),
(3018, 652415511, 888342),
(3019, 652415511, 888341);
```

SQL available at

database page

https://www.alps.academy/create-a-simple-database/

junction table page (create registration / insert data)

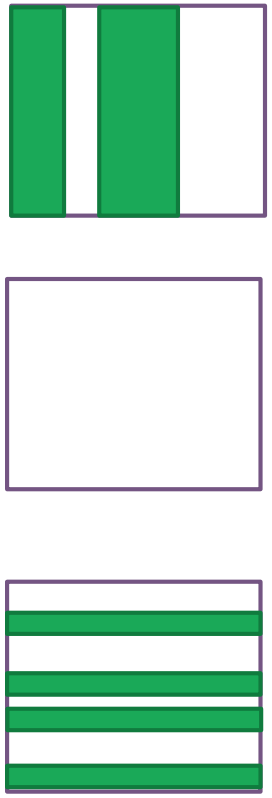
https://www.alps.academy/junction-table/

more SQL page (SQL select statements)

https://www.alps.academy/sql-multiple-where-clause/

Using or querying Database data from multiple tables

SQL - DML



SELECT	what you want to see
FROM	from where
WHERE	join tables
AND	

alias on tables

```
select student_name
from student s, registration r
where s.student_id = r.student_id
```

use alias s for the student table

use alias r for the registration table

saves retyping the table name

alias on tables

```
select s.student_id
from student s, registration r
where s.student_id = r.student_id
```

s.student_id as
there are two student_id
in both tables
so SQL doesn’t know which one you want
so you have to say which one

student on courses

```
select student_name
from student s, registration r
where s.student_id = r.student_id
```

student on courses (duplicates removed)

```
select distinct student_name
from student s, registration r
where s.student_id = r.student_id
```

courses with students

```
select c.course_code
from course c, registration r
where c.course_code = r.course_code
```

courses with students (duplicate removed)

```
select distinct c.course_code, course_name, course_date
from course c, registration r
where c.course_code = r.course_code
```

later - join

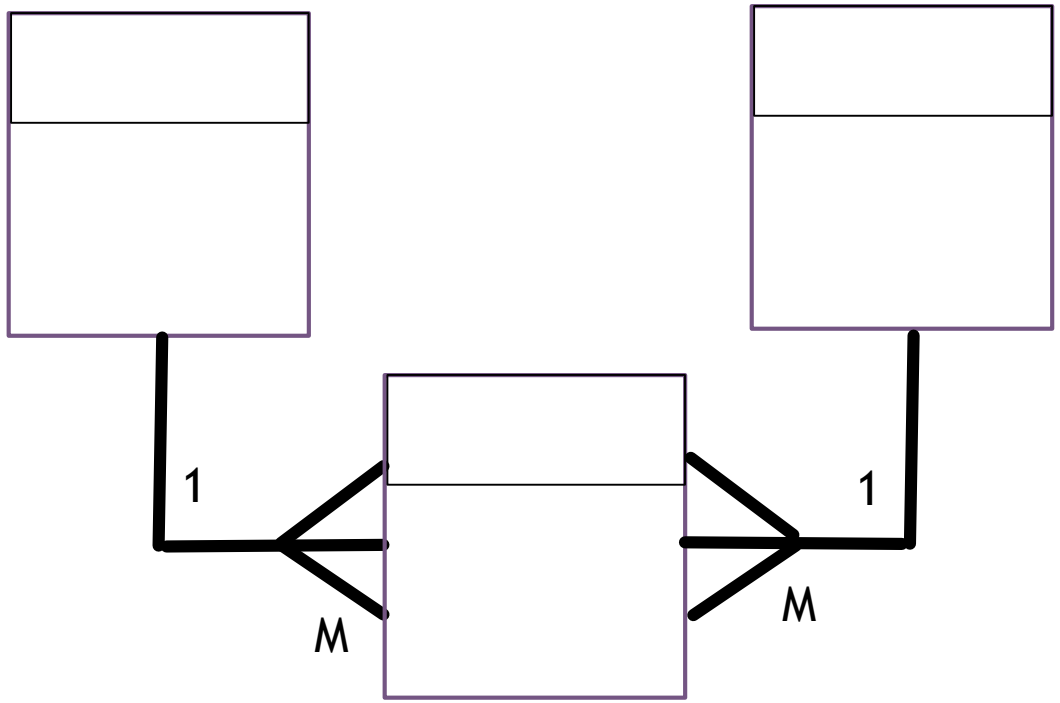
```
-- Use table aliases for shorter references
SELECT s.student_id, s.student_name, r.course_code
FROM student AS s
JOIN registration AS r ON s.student_id = r.student_id;
```

multiple table

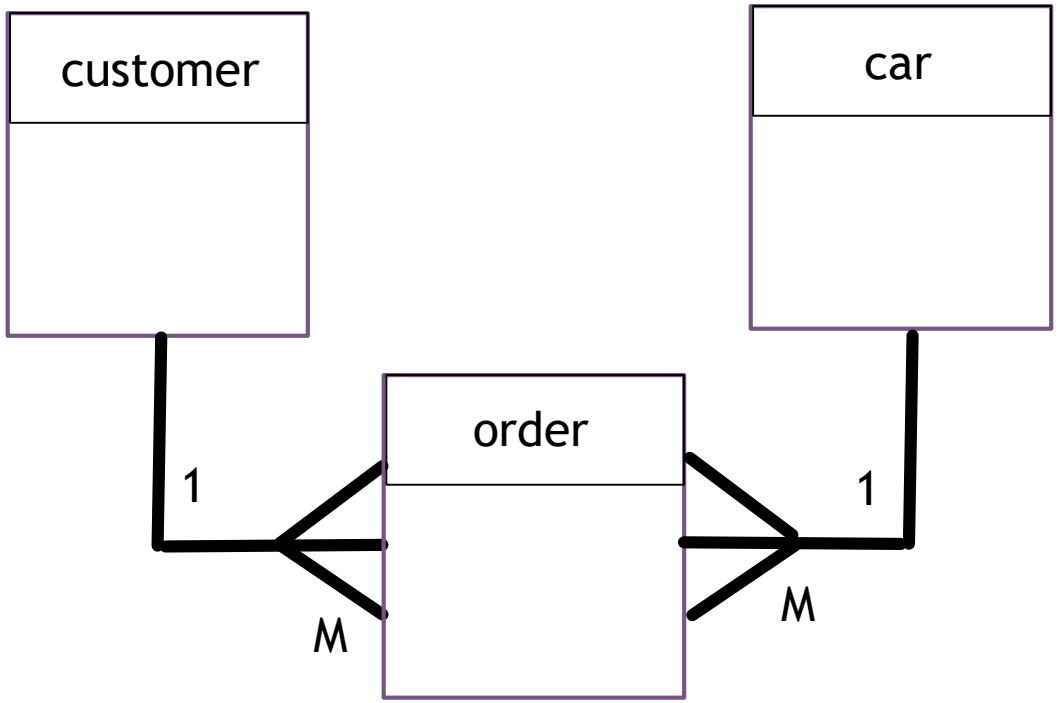
is the design ok ?

design considerations

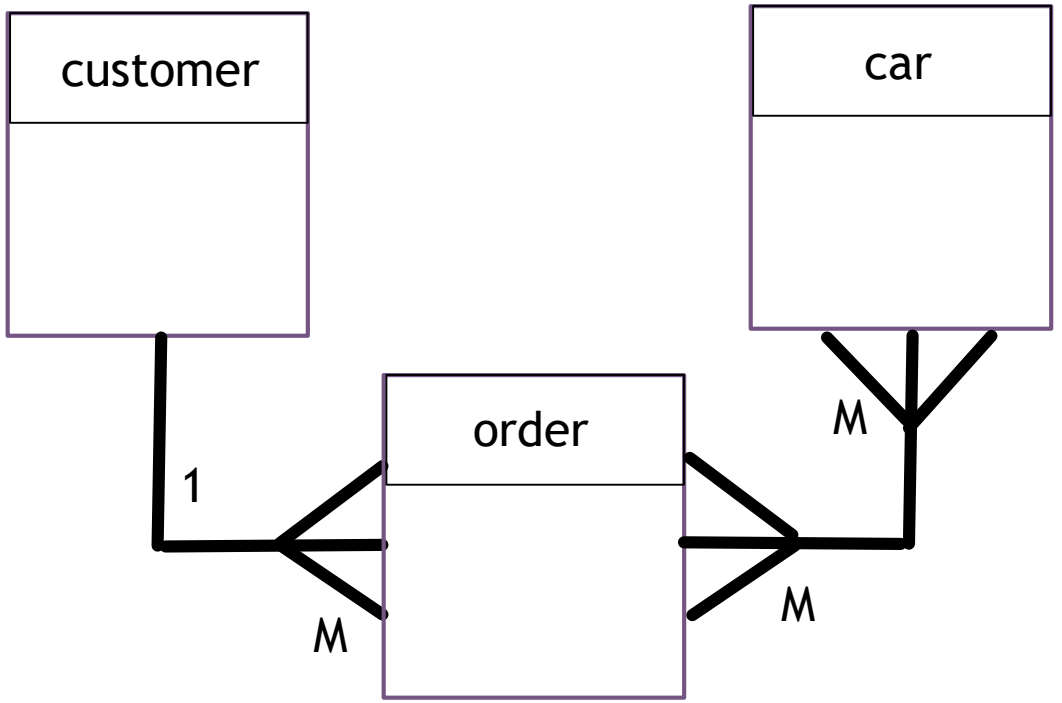
design



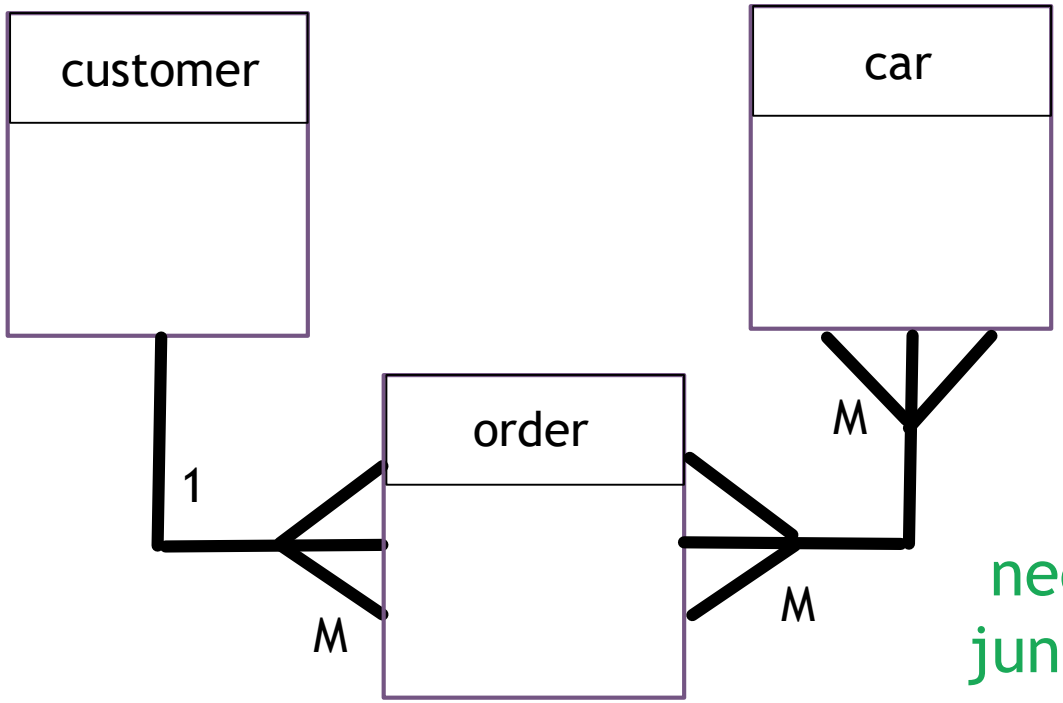
one car one order



multiple cars one order

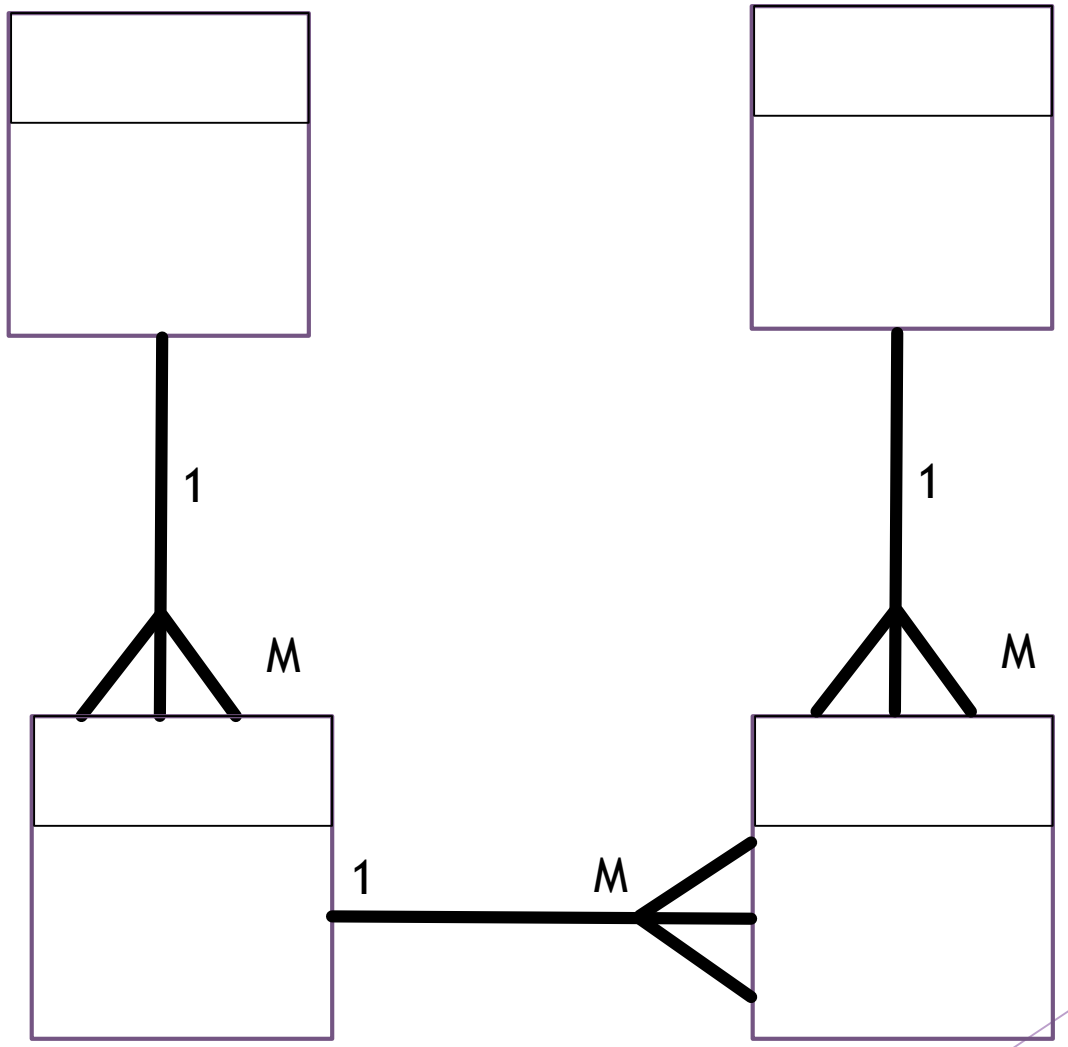


multiple cars one order



need a junction table

design



receipt (order)

order

item_order

item_order

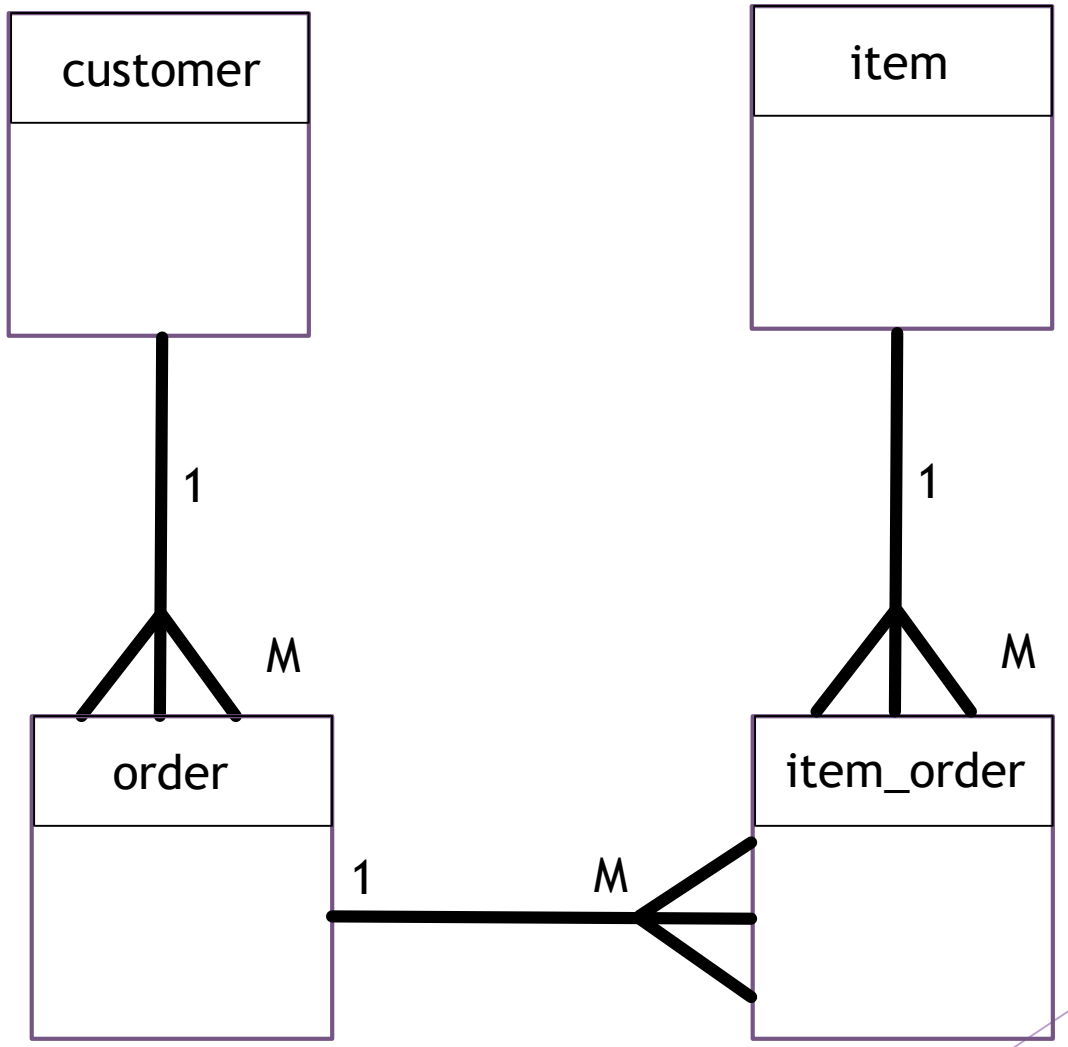
item_order

item_order

item name	item price	item amount	cost of items
item name	item price	item amount	cost of items
item name	item price	item amount	cost of items
item name	item price	item amount	cost of items

TOTAL ORDER COST

sell multiple items on an order



More SQL
from multiple tables

using AND

```
select student_name, course_code
from student s, registration r
where s.student_id = r.student_id
and s.student_id = 652415503
```

using AND

```
select c.course_code, course_name, course_date, r.student_id
from course c, registration r
where c.course_code = r.course_code
and c.course_code = 888342
```

multiple tables and multiple ANDs

```
select c.course_code, course_name, course_date, r.student_id,
student_name
from course c, registration r, student s
where c.course_code = r.course_code
and c.course_code = 888342
and s.student_id = r.student_id
order by s.student_id
```

students on courses with multiple order by

```
select c.course_code, course_name, course_date, r.student_id,
student_name
from course c, registration r, student s
where c.course_code = r.course_code
and s.student_id = r.student_id
order by c.course_date, c.course_code,s.student_id
```

students on which courses

```
select r.student_id, student_name, c.course_code, course_name,
course_date
from course c, registration r, student s
where c.course_code = r.course_code
and s.student_id = r.student_id
order by s.student_id, c.course_date, c.course_code
```

aggregate SQL - count

```
SELECT COUNT(s.student_id) AS student_count
FROM registration r, student s
WHERE r.student_id = s.student_id
```

group by and having

```
SELECT c.course_name, c.course_date, COUNT(s.student_id) AS
student_count
FROM registration r, student s, course c
WHERE r.course_code = c.course_code
AND r.student_id = s.student_id
GROUP BY c.course_name, c.course_date
HAVING COUNT(s.student_id) > 0
ORDER BY c.course_date, c.course_name;
```

group by and having

```
SELECT s.student_name, COUNT(r.course_code) AS course_count
FROM registration r, student s
WHERE r.student_id = s.student_id
GROUP BY s.student_name
HAVING COUNT(r.course_code) > 0
ORDER BY s.student_name;
```

insert, update, delete

```
INSERT INTO course (course_code, course_name, course_date) VALUES
(888123, 'Data Analysis Basics', '2024-02’);

UPDATE course SET course_date = '2024-2' WHERE course_code = 888123;

DELETE FROM course WHERE course_code = 888123;
```

Database design: normalization

level 1 (1 NF)

columns must have

- data with only one value
- be unique with different names
- contain the same data type
- data in any order
- rows that can be uniquely identified

level 2 (2 NF)

columns must

- conform to level 1 (1 NF)
- be dependent on the primary key

more simple

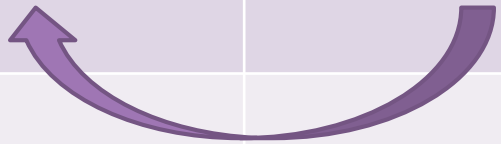
improves integrity

reduces redundancy

level 3 (3 NF)

columns must not contain a transitive dependency

student_id	student_name	course_id	course_name



course name is dependent on course id not student id

therefore it is a transitive dependency

level 4 & 5 (4 NF & 5NF)

can it be broken down more?

department	course	instructor
ICDI1	888001	Dr M
ICDI2	888002	Dr A
ICDI1	888001	Dr M
ICDI3	888003	Dr T

level 4 & 5 (4 NF & 5NF)

can it be broken down more?

department	instructor
ICDI1	Dr M
ICDI2	Dr A
ICDI1	Dr M
ICDI3	Dr T

department	course
ICDI1	888001
ICDI2	888002
ICDI1	888004
ICDI3	888003

course	instructor
888001	Dr M
888002	Dr A
888004	Dr M
888003	Dr T

Thank you!

any questions?