Exam 2 Review

Lecture 13

Format & Process

- 1.5 hours (during class, Thursday 3/20)
  - Your time will start shortly after class begins
- 4 parts
  - Normalization: Relations + FD -> analysis + decomp
  - Conceptual: T/F ERD interpretation, narrative -> ERD
  - Logical: ERD -> relations
  - Physical: description -> index? + why
  - Bonus: FDs -> key, state + FD -> holds/invalidation(s)
- Free to use: 1 front/back 8x11”
  - I will supply ERD reference sheet, extra blank paper
  - Any violation of academic integrity will result in failing the class and a report to the University
- I will be in-person
  - And we’ll debrief shortly after it’s done :)

Content

Database design…

- Normalization
- Conceptual (i.e., ER Diagrams)
- Logical (i.e., mapping ERDs -> relations)
- Physical (indexing)

Normalization

- What are the goals of normalization?
  - Spurious tuples? Additive decomposition?
  - Modification anomalies? Examples!
- Functional dependencies
  - Definition, relationship to keys
  - Do they hold given data? If not, violating pair(s)
  - Trivial, transitive, full
- Normal forms
  - What do 1NF/2NF/3NF require?
  - Decomposition algorithm

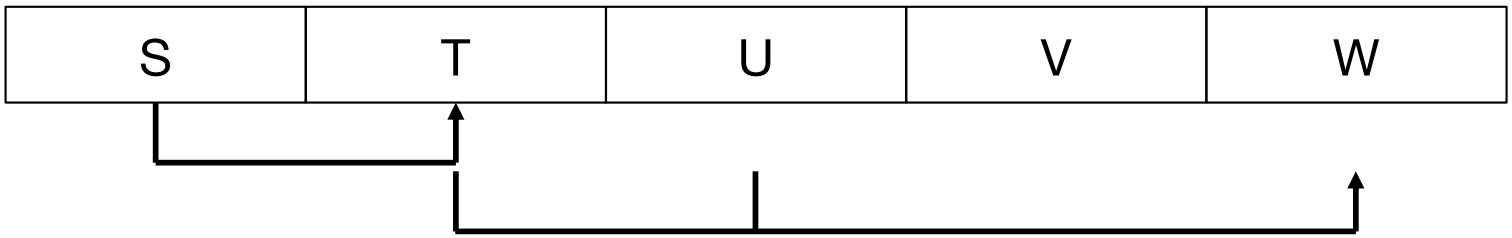
Exercise

Non-Trivial FDs

- S -> T
- TU -> W

Candidate Key(s)

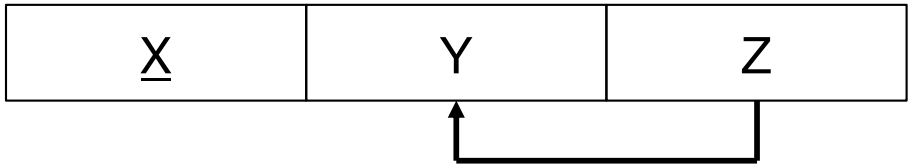
- SUV



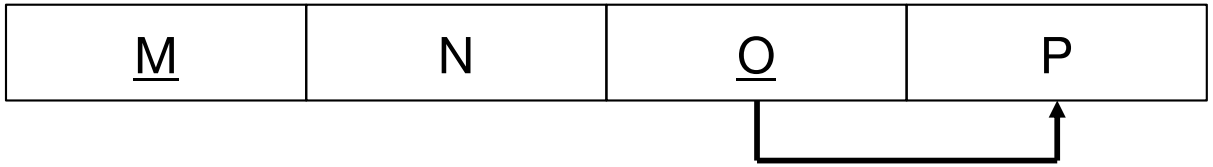
Exercise

Which NF? Why? Decompose to 3NF.

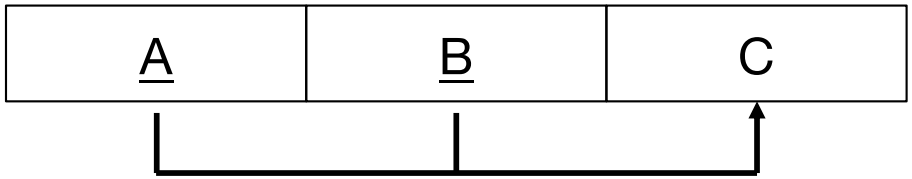
Foo



Bar

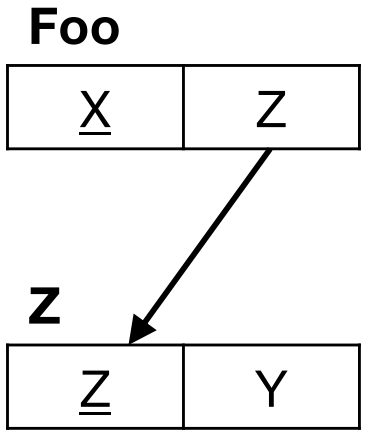


Baz

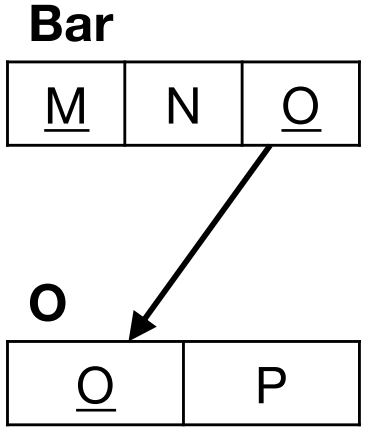


Answer (1)

- Foo is in 2NF
  - 2NF (single PK attr)
  - Y is tFD on PK
  - Post:
    - Foo/Z: single PK/np

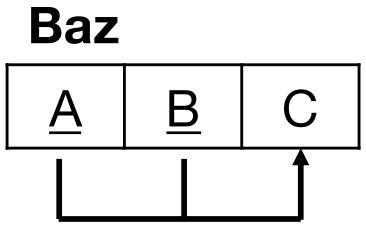


- Bar is in 1NF
  - P is not fFD on PK
  - Post:
    - Bar: N fFD on PK, single np
    - O: single PK/np



Answer (2)

- Baz is in 3NF
  - 2NF: C is fFD on PK
  - 3NF: single np

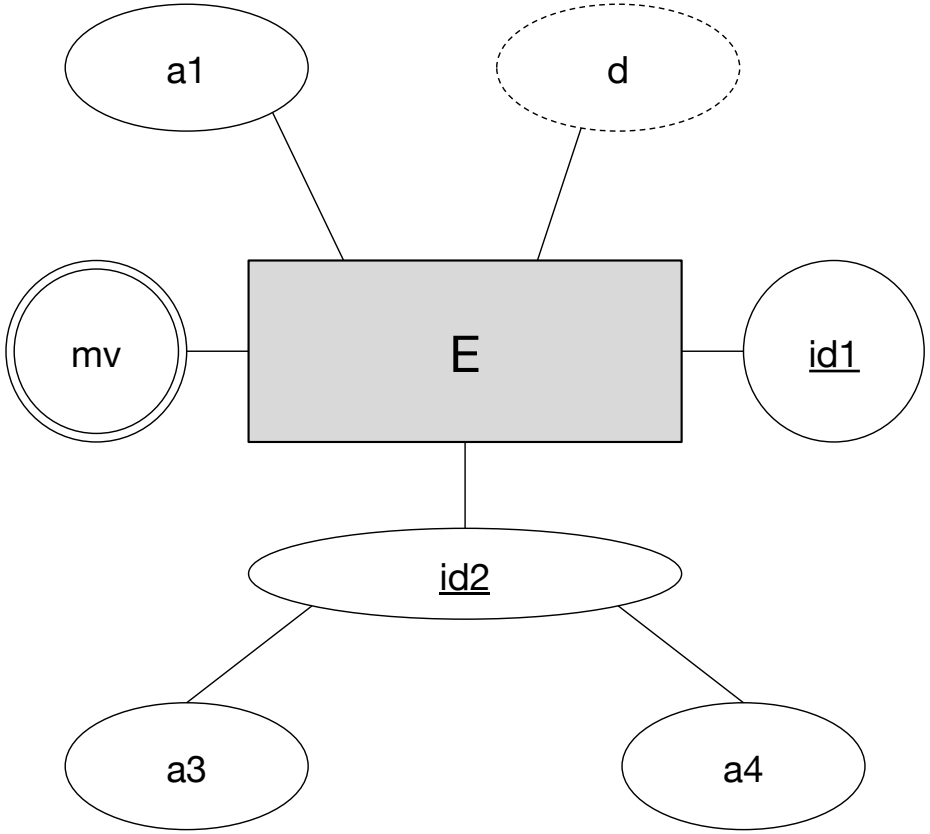


ER Diagrams & Mapping

- Conceptual design: goals, approaches
- All the notation we covered
  - Entities: weak/strong
  - Attributes: composite, multi-valued, derived, keys
  - Relationships: cardinality, structural, attributes
  - Specialization/Generalization
  - When to use!
- Mapping to tables
  - Multiple methods for specialization/generalization

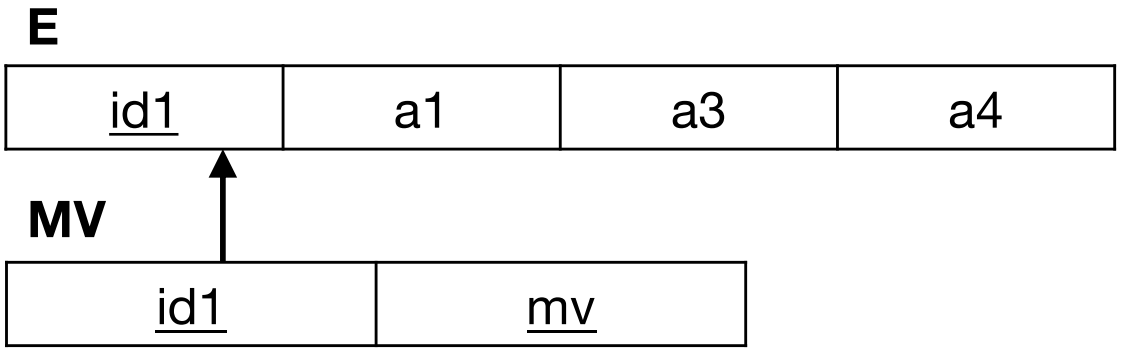
Exercise

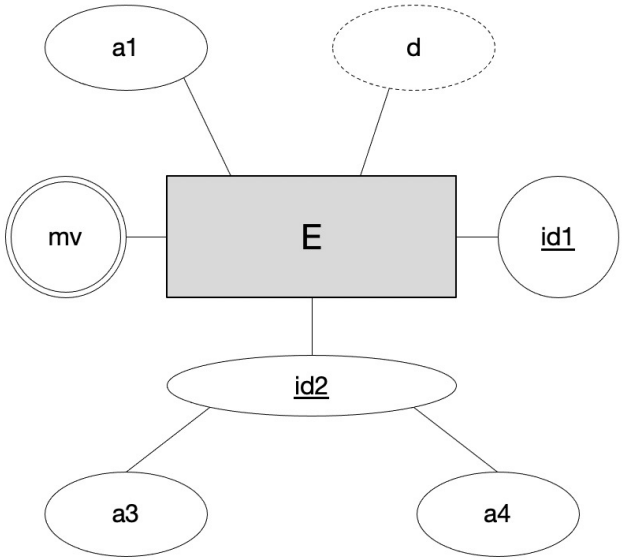
- Describe in words the following ERD
  - How can you identify an instance of *E*?
- Map E to relation(s)
  - What are the primary key(s)?
  - What happens to other key(s)?



Answer

- All E's have an a1, an id1, an id2 composed of a3 and a4, and some number of mv's. By combining these you can determine the E's d.
- An E can be uniquely identified by either its id1, or the combination of a3 and a4.



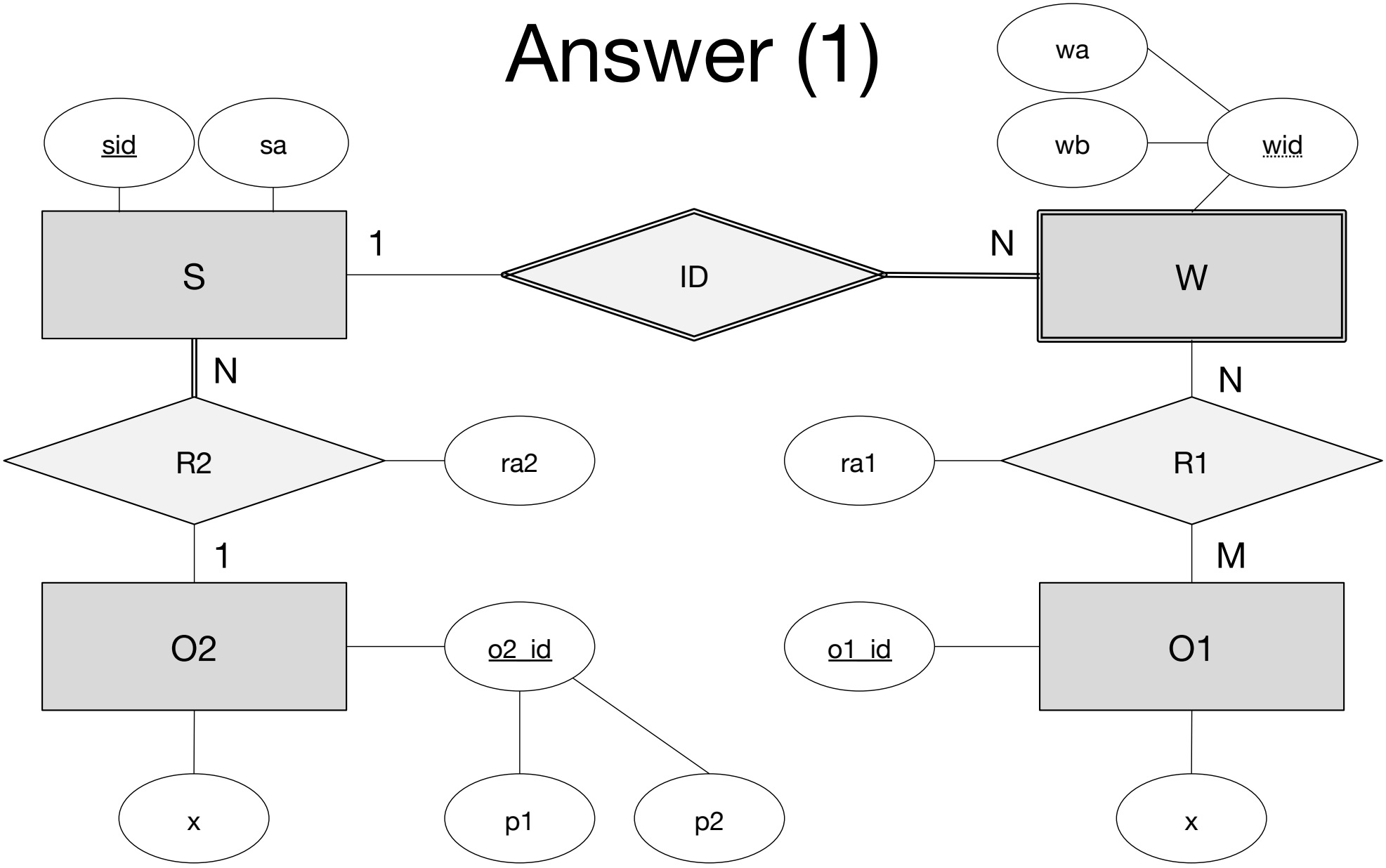


Exercise

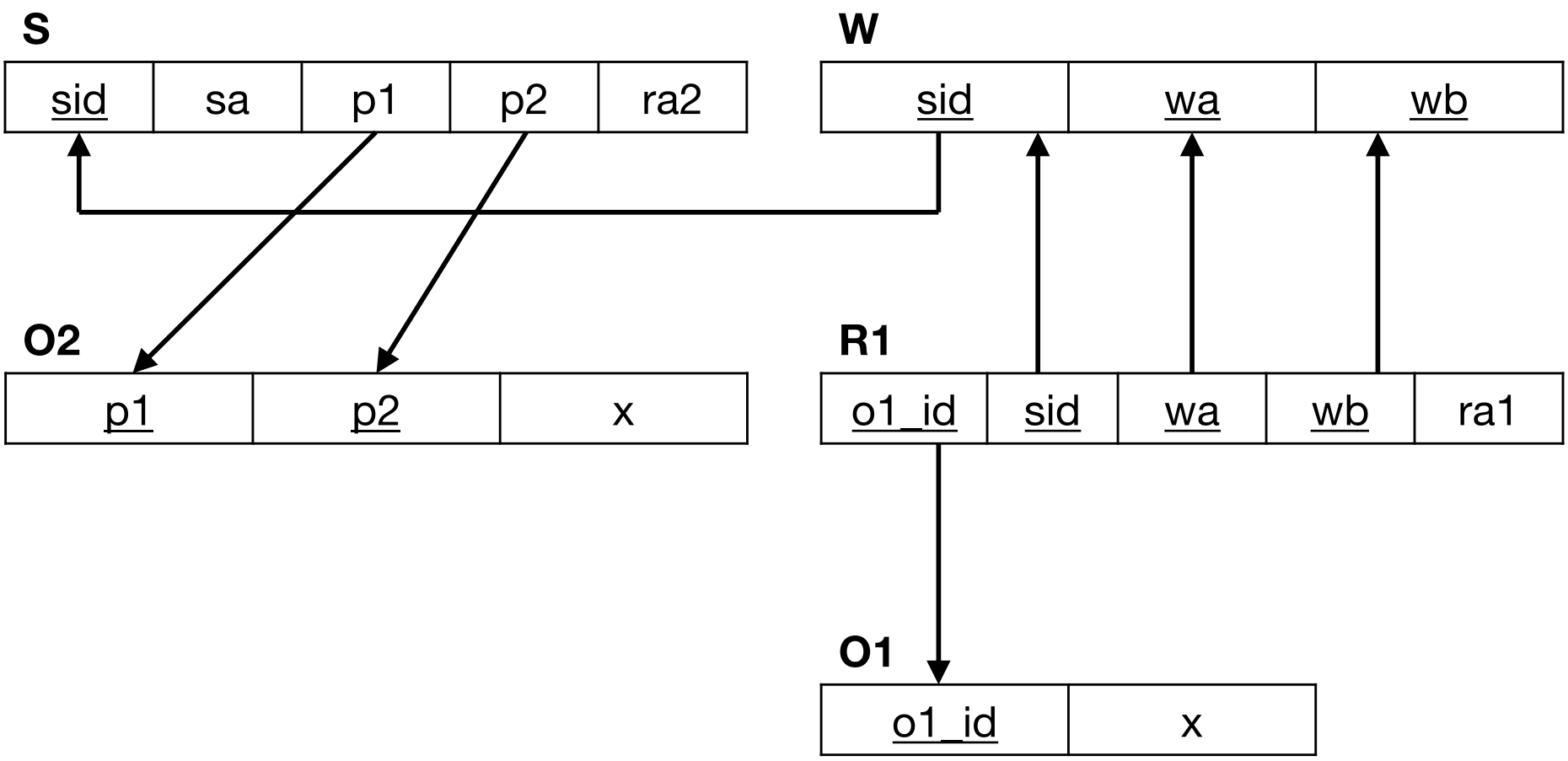
Produce an ERD & corresponding relational schema

- An **S** has an **sa** and can be uniquely identified by its **sid**
- Each **S** must **R2** with a single **O2**, whereas each **O2** may **R2** with any number of **S**'s. When an **S** **R2**'s an **O2**, it is important to note the corresponding **ra2**
- An **O2** has an **x** and can be uniquely identified by its **o2_id**, which is comprised of **p1** and **p2**
- A **W** is identified by its corresponding **S**, in combination with its own **wid**, consisting of a **wa** and **wb**
- Each **W** can **R1** with any number of **O1**'s, and likewise each **O1** can **R1** with any number of **W**'s. Each **R1** interaction has a corresponding **ra1**
- An **O1** is uniquely identified by its **o1_id** and also has an **x**

Answer (1)

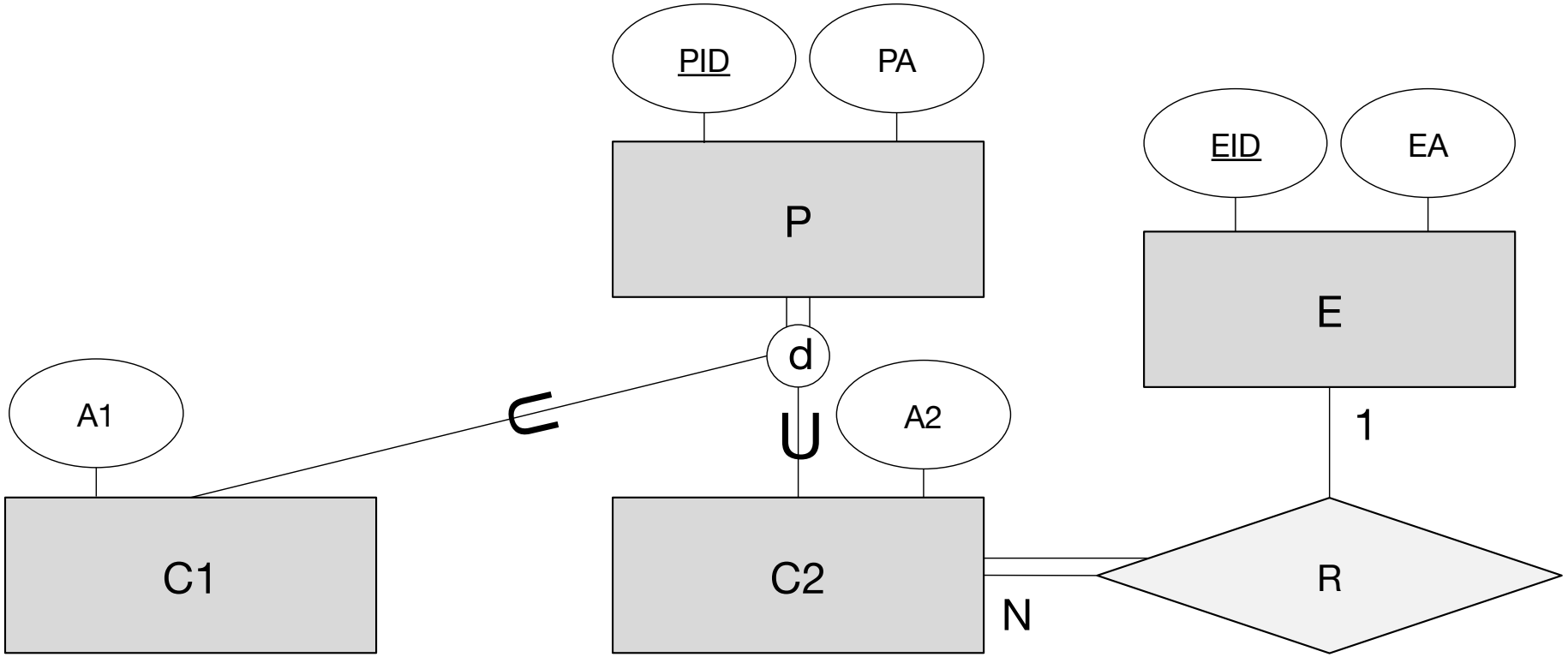

wa
wb
wid
sid
sa
S
1
ID
N
W
N
N
R2
ra2
ra1
R1
1
M
O2
o2_id
o1_id
O1
x
p1
p2
x

Answer (2)

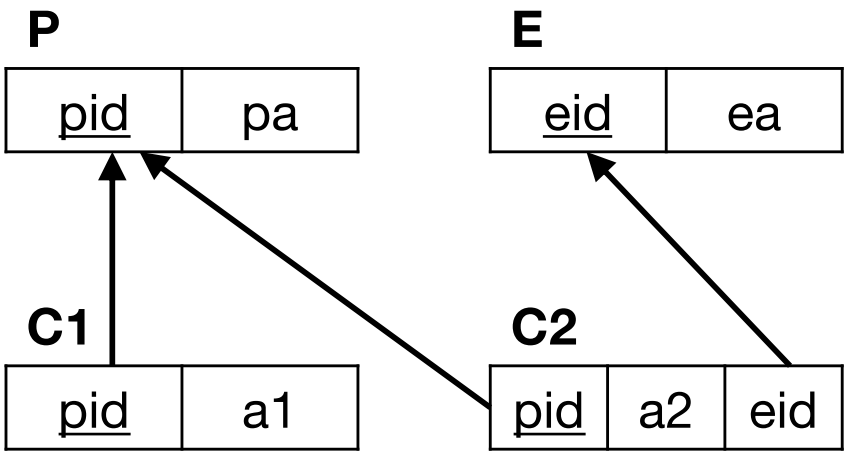

S
sid
sa
p1
p2
ra2
W
sid
wa
wb
O2
p1
p2
x
R1
o1_id
sid
wa
wb
ra1
O1
o1_id
x

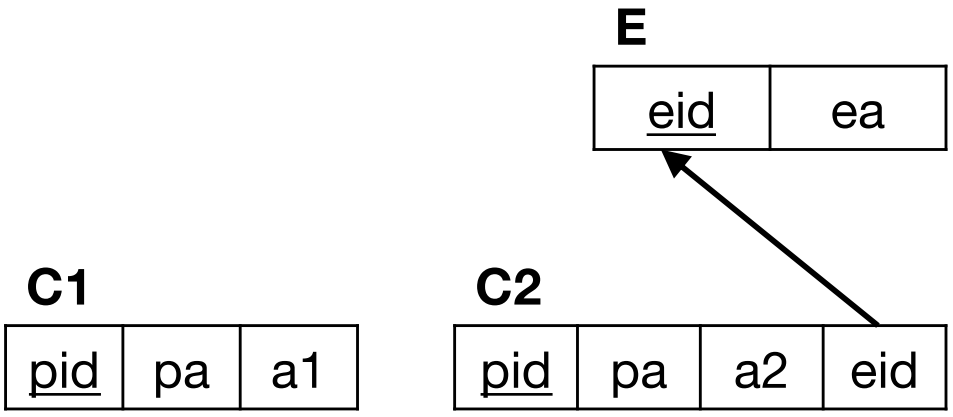
Exercise

Map this ERD to relations in **two** different ways. Pros and cons of each?



Answer


P
pid
pa
E
eid
ea
C1
pid
a1
C2
pid
a2
eid


E
eid
ea
C1
pid
pa
a1
C2
pid
pa
a2
eid

Indexing

- What is an index? What are the potential costs & benefits of using one?
- What factors should be considered when choosing whether or not to use an index?
- What are clustered indexes? Covering?
- Why use a hash table vs b+-tree?

Exercise

Consider the parameterized query on relation `T1(X, Y)`:

```
SELECT *
FROM T1
WHERE Y>?
ORDER BY X DESC;
```

Assume the query currently runs too slowly.

To optimize the performance of this query, is it better to use an index based upon a B+-tree or a hash table? Why?

Answer

Consider the parameterized query on relation `T1(X, Y)`:

```
SELECT *
FROM T1
WHERE Y>?
ORDER BY X DESC;
```

Assume the query currently runs too slowly.

To optimize the performance of this query, is it better to use an index based upon a B+-tree or a hash table? Why?

B+-tree: due to range & ordering

Exercise

You have been tasked to make a voting app FAST. It's functionality: a user (via their ID) votes yes/no (or updates their prior vote) and then later we count the results.

The table in question: `T2(UID, vote)`

1. What additional information should you request before considering indexing?
2. What index(es) should you create? Why?

Answer

1. How many users? Relative frequency of votes vs tallying?

2. None: ID likely has clustered index as PK; vote has few distinct values, potential for lots of updates.

Good Luck on Exam 2 :)