COMP2411

Tutorial 3 (Week 7)

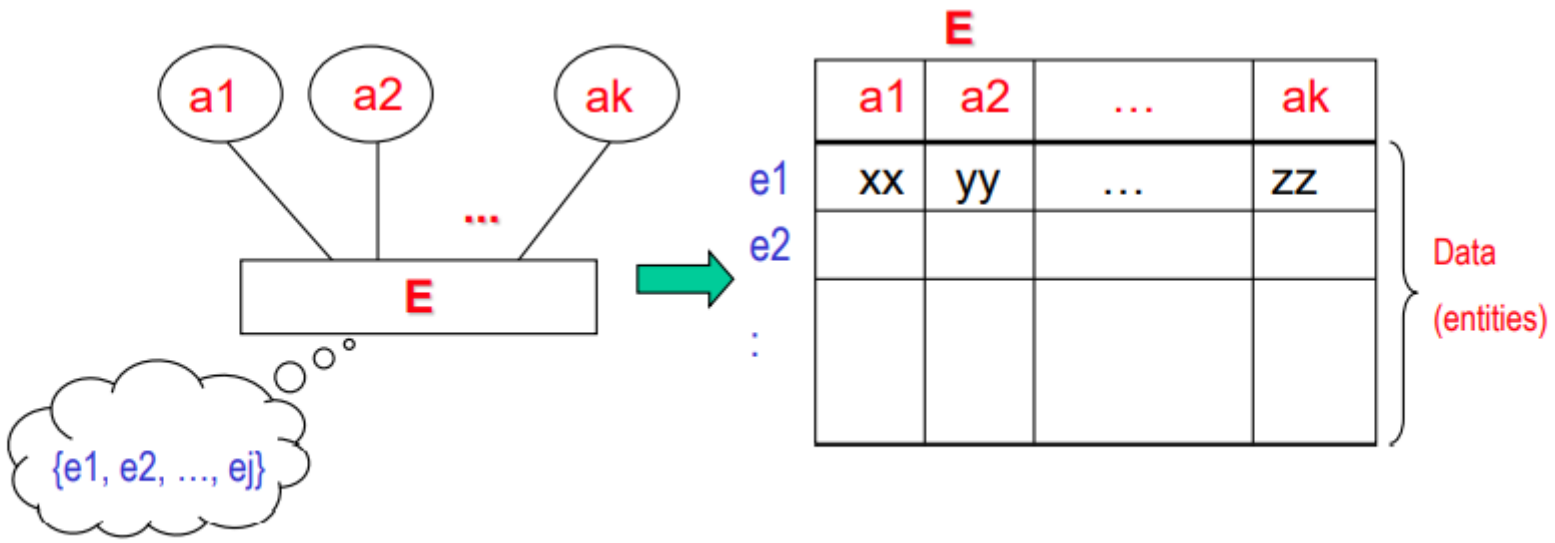
Q1.

1. For the chosen application you had modeled using ER during the tutorial class last time (*in Week 3*), please convert the ER diagram into relational schema (*table structures*) by following the "rules" described in the lecturing class.

Q1.

ER vs. Relational Data Model

- Mapping ER Diagrams into Tables
  - Representation of (Strong) Entity Sets



Page 27, Lectures 2-3.

Q1.

9.1 Relational Database Design Using ER-to-Relational Mapping

9.1.1 ER-to-Relational Mapping Algorithm

In this section we describe the steps of an algorithm for ER-to-relational mapping. We use the COMPANY database example to illustrate the mapping procedure. The COMPANY ER schema is shown again in Figure 9.1, and the corresponding COMPANY relational database schema is shown in Figure 9.2 to illustrate the map-

Elmasri, R. & Navathe, S.B. (2011). *Fundamentals of Database Systems: Chapter 9 Relational Database Design Using ER-to-Relation Mapping*

Q2.

2. Consider the following SQL query:

```
SELECT R.a1, R.a2
FROM   R, R1, R2
WHERE  R.a1 = R1.a1
AND    R.a1 = R2.a1;
```

Under what situations does the above query select tuples of the form (`R.a1, R.a2`) when `R.a1` appears in both `R1` and `R2`?

(***Hint***: *Examine carefully the cases where* `R1` *and/or* `R2` *may contain empty/null data.*)

Q2.

R

a1	a2
1	4
	5
2	
3	6

R1

a1	a3
1	7
2	
	8
3	9

R2

a1	a4
1	10
2	
	12
3	13

Q3.

3. A database is to be set up to maintain the pool of lecture theatres and to assist in their allocation to courses. Consider the following relation/table with the set of functional dependencies **F** defined on its attributes:

```
CourseRmAlloc(CourseId, CourseName, Year, Lecturer, Enrollment, RoomId,
              RoomCapacity, Day, Time)

F = { CourseId -> CourseName,          CourseName -> CourseId,
      CourseId, Year -> Lecturer,      CourseId, Year -> Enrollment,
      RoomId -> RoomCapacity,          RoomId, Year, Day, Time -> CourseId,
      CourseId, Year, Day, Time -> RoomId }
```

Q3.

a) Find all the candidate keys of `CourseRmAlloc`. Demonstrate that they are indeed candidate keys.

b) Determine the highest normal form that the relation `CourseRmAlloc` is in and, justify your answer. What problems will arise with this relation?

c) Considering the following decomposition, give all the candidate keys for the relations `Course` and `RoomAlloc`. State what normal form each relation is in.

```
Course(CourseId, CourseName, Year, Lecturer, Enrollment)
RoomAlloc(RoomId, RoomCapacity, Day, Time, CourseId)
```

Q3.

```
CourseRmAlloc(CourseId, CourseName, Year, Lecturer, Enrollment, RoomId,
              RoomCapacity, Day, Time)

F = { CourseId -> CourseName,          CourseName -> CourseId,
      CourseId, Year -> Lecturer,     CourseId, Year -> Enrollment,
      RoomId -> RoomCapacity,         RoomId, Year, Day, Time -> CourseId,
      CourseId, Year, Day, Time -> RoomId }
```

a) Find all the candidate keys of `CourseRmAlloc`. Demonstrate that they are indeed candidate keys.

b) Determine the highest normal form that the relation `CourseRmAlloc` is in and, justify your answer. What problems will arise with this relation?

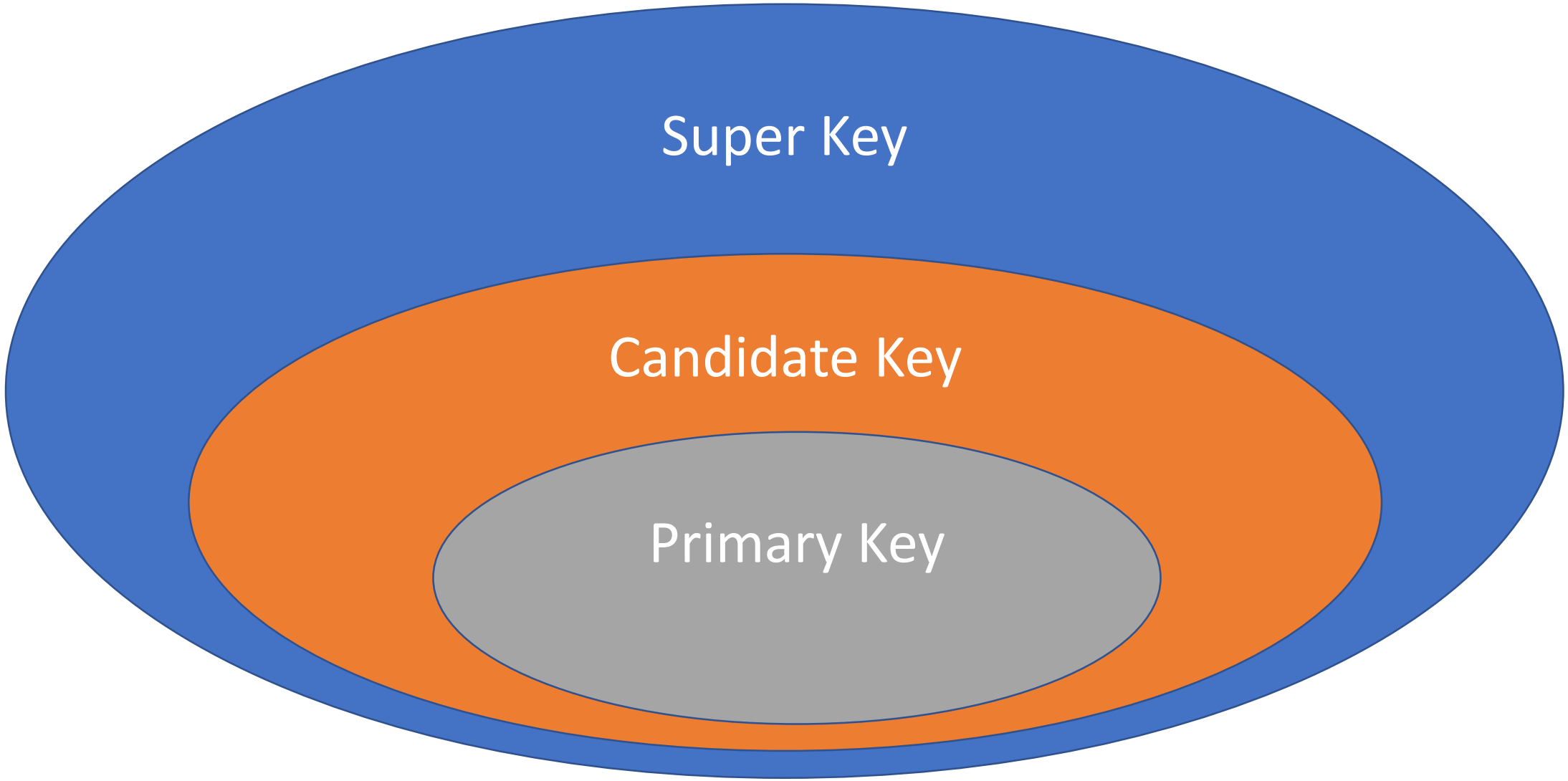
Inference Rules for FDs

- Given a set of FDs $\mathbb{F}$, we can **infer** additional FDs that hold whenever the FDs in $\mathbb{F}$ hold
- Armstrong's inference rules:
  - IR1. (**Reflexive**) If Y is a *subset of* X, then X → Y
  - IR2. (**Augmentation**) If X → Y, then XZ → YZ
    - (Notation: XZ stands for X U Z)
  - IR3. (**Transitive**) If X → Y and Y → Z, then X → Z

- IR1, IR2, IR3 form a **sound** and **complete** set of inference rules
  - Sound: These rules are true
  - Complete: All the other rules that are true can be deduced from these rules

Integrity Constraints

- **Key** constraints
  - **Superkey** of R: A set of attributes SK of R such that no two tuples *in any valid relation instance r(R)* will have the same value for SK. That is, for any distinct tuples t1 and t2 in r(R), t1[SK] **<>** t2[SK].
  - **Candidate Key** of R: A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.
  - **Primary Key** of R: choice by the DB designer when there are more than

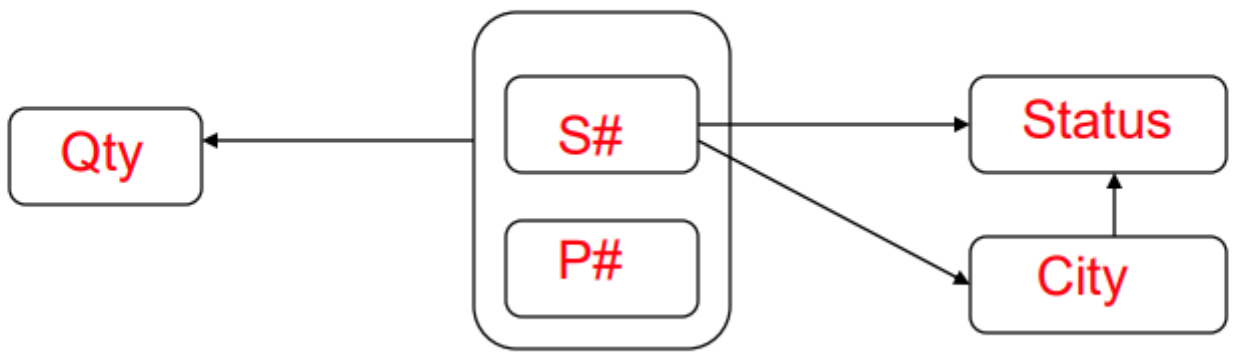

Super Key
Candidate Key
Primary Key

Relational DB Design

- First Normal Form (1NF)
  - a relation R is in 1NF *if and only if* all underlying domains contain ***atomic values*** only
  - Example:

    FIRST(S#, Status, City, P#, Qty)

    and its Functional Dependency Diagram:

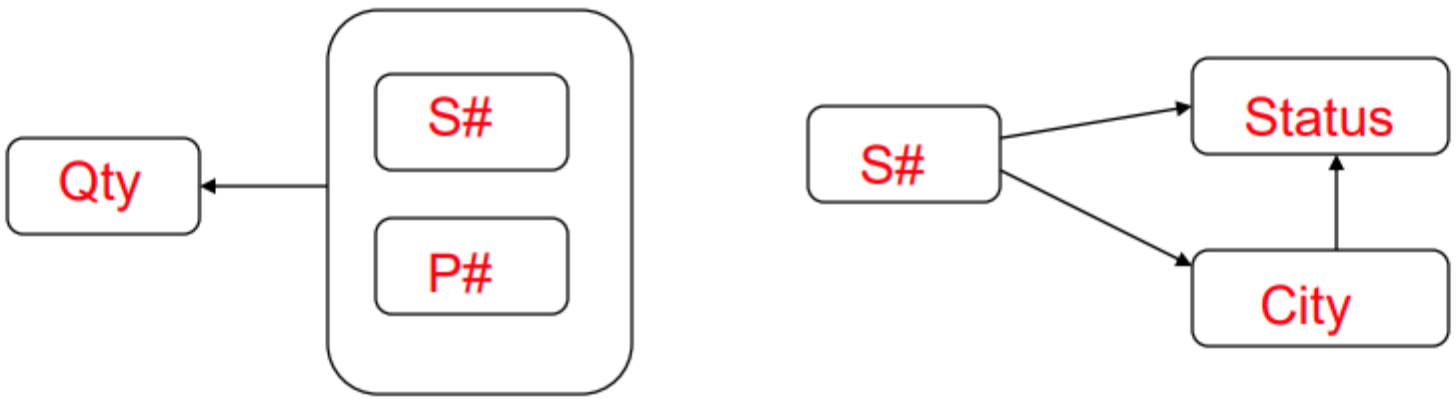


What's the primary key?

Relational DB Design

- **Definition**: A relation R is in 2NF *if and only if* it is in 1NF and every non-key attribute is **fully** dependent on any candidate key.

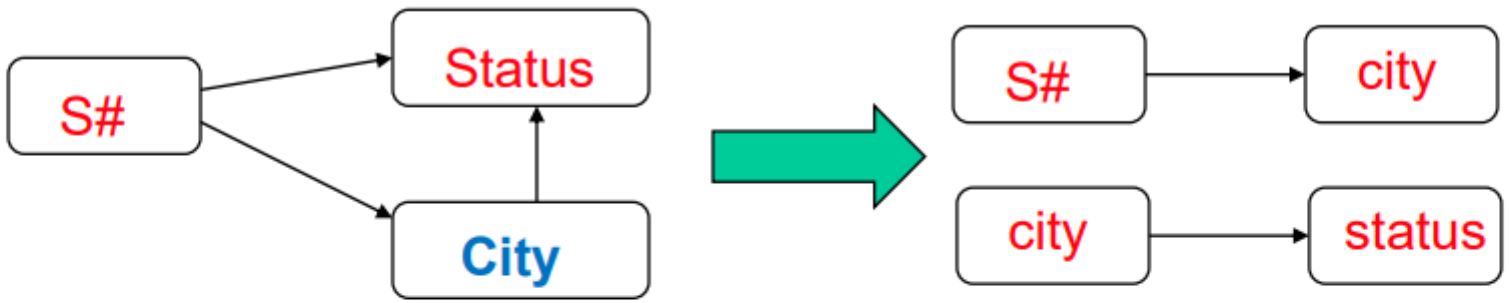
So, both relations are in 2NF.



Relational DB Design

- Now the undesirable problems of 2NF seem to be gone
- **Definition**: A relation R is in 3NF *if and only if* it is in 2NF and every non-key attribute is **non-transitively** dependent on any candidate key.

So, in the above case, all relations are now in 3NF.



- In general, 3NF is desirable and powerful enough
- But, still there are cases where a stronger normal form is needed

Q3.

```
CourseRmAlloc(CourseId, CourseName, Year, Lecturer, Enrollment, RoomId,
              RoomCapacity, Day, Time)

F = { CourseId -> CourseName,          CourseName -> CourseId,
      CourseId, Year -> Lecturer,     CourseId, Year -> Enrollment,
      RoomId -> RoomCapacity,         RoomId, Year, Day, Time -> CourseId,
      CourseId, Year, Day, Time -> RoomId }
```

c) Considering the following decomposition, give all the candidate keys for the relations `Course` and `RoomAlloc`. State what normal form each relation is in.

```
Course(CourseId, CourseName, Year, Lecturer, Enrollment)
RoomAlloc(RoomId, RoomCapacity, Day, Time, CourseId)
```

Try this one before you look at the suggested solution.