Normalization

- Model design is an intuitive process
  - Group attributes into entity types
  - Relate entity types to each other
- Many models can fulfill the same set of requirements for data retrieval

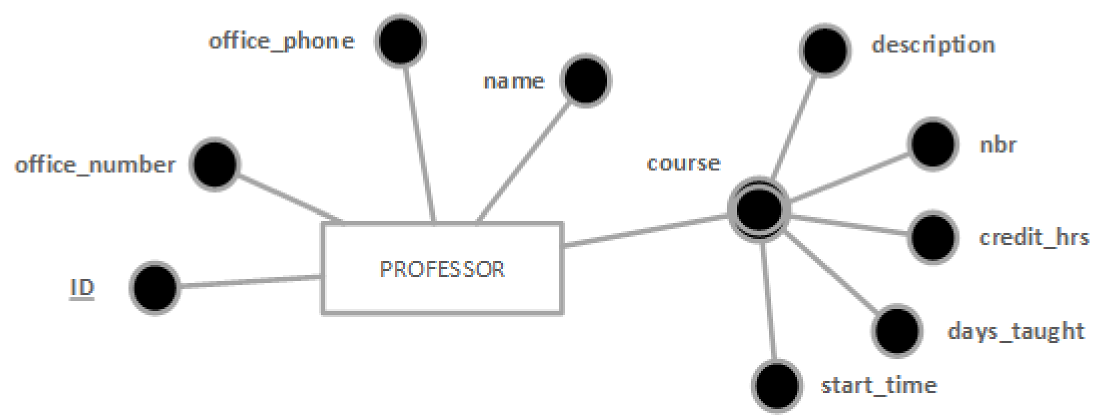
Normalization

* Model needs to support ALL operations expected on a database
    * CRUD
* Side effects (or anomalies) can easily occur when a system tried to modify data in the database
* These anomalies will impact *every* system that works with the database

Modification Anomalies

- Modification anomalies
  - Insert anomalies
  - Update anomalies
  - Delete anomalies
- You want a database design that minimizes the potential for modification anomalies

Example



ID	NAME	OFFICE_NBR	PHONE_NBR	COURSE_NBR	DESCRIPTION	CREDIT HOURS	DAYS	TIMES
1881	Dr Green	2142	6-7164	CGS1060	Intro to Microcomputers	4	TR	14:00
				COP1334	C++ Programming	4	TR	8:00
				COP2800	Java Programming	4	TR	10:30
2142	Dr Gonzalez	2111	6-5543	CGS1540	Database Concepts and Design	4	M	17:00
766	Dr Pierre	2144	6-7160	CGS1060	Intro to Microcomputers	4	MWF	10:00
641	Ms Osplant	2146	6-7140	COP2800	Java Programming	4	MWF	11:00
				COP2805	Adv Java Programming	4	MWF	15:30
2004	Dr Chen	2150	6-7141	CGS1060	Intro to Microcomputers	4	W	17:00

What are the problems with the professor table?

- If a course description is changed, the change must be made multiple times – Update Anomaly
- If we delete a professor, we lose course information – Delete Anomaly
- A new course cannot be in database until a professor is identified – Insert Anomaly
- *Chapter 7, section 7.1 has comprehensive example*

Causes of Anomalies

- Data redundancy
  The description of course CGS1060 is repeated many times

- Missed entity types
  An OFFICE and COURSE entity types are probably needed

- Misplaced attributes
  Professor’s phone number might be an OFFICE attribute, not a PROFESSOR attribute

How to minimize modification anomalies

- Make sure every entity type captures a different, single theme
- This will lead to splitting up entity types into entity types with fewer attributes in each
- This process is called ***Normalization***

functional dependencies

* Before diving into **normalization,** we need to understand **functional dependencies**

* **Definition:**

  A functional dependency is constraint in a **relation schema** of the form

  A → B

  where a value of attribute B is dependent on the value of attribute A

  A is called the determinant
  B is called the dependent

  A determines **B**
  B is dependent on **A**

functional dependencies

- The initial set of functional dependencies come from the business rules
  - Expressed in normal language
  - If not refined they often have redundancies and contradictions
- These functional dependencies have to be refined to the minimum necessary set

Functional Dependency Inference Rules

* You can refine functional dependencies using Armstrong's Axioms (in page 368 in text)

Table 7.1 Inference rules for functional dependencies: Armstrong's axioms

Rule	Definition
Reflexivity	If Y is a subset of X [i.e., if X is (A,B,C,D) and Y is (A,C)], then X → Y. (The reflexivity rule defines trivial dependency as a dependency that is impossible to *not* satisfy.)
Augmentation	If X → Y, then {X,Z} → {Y,Z}; also, {X,Z} → Y.
Transitivity	If X → Y, and Y → Z, then X → Z.
Decomposition	If X → {Y,Z}, then X → Y and X → Z.
Union (or additive)	If X → Y, and X → Z, then X → {Y,Z}.
Composition	If X → Y, and Z → W, then {X,Z} → {Y,W}.
Pseudotransitivity	If X → Y, and {Y,W} → Z, then {X,W} → Z.
References: Armstrong, W. W. "Dependence Structures of Data Base Relationships" *Proc. IFIP Congress*, Stockholm, Sweden (1974); Darwen, H. "The Role of Functional Dependencies in Query Decomposition," In C.J. Date and H. Darwen, *Relational Database Writings 1989 – 1991*, Addison-Wesley (1992).	

Normal Forms

- Normalization is the process of converting a relation from one normal form (NF) to a higher normal form (NF)
- Normal forms of a relation is determined by all the functional dependencies between its attributes
- There are many NFs
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)
  - Domain-Key Normal Form (DKNF)
  - And more

Normal Forms

- A best practice for a transactional database is that all tables conform to 3NF or BCNF

Normal Forms

First Normal Form (1NF):

A relation schema is in First Normal Form when all of its attributes are atomic and single-valued.

- no composite or molecular attributes
- no multi-valued attributes
- *sound familiar?*
  - a relational schema, by definition, is in 1NF

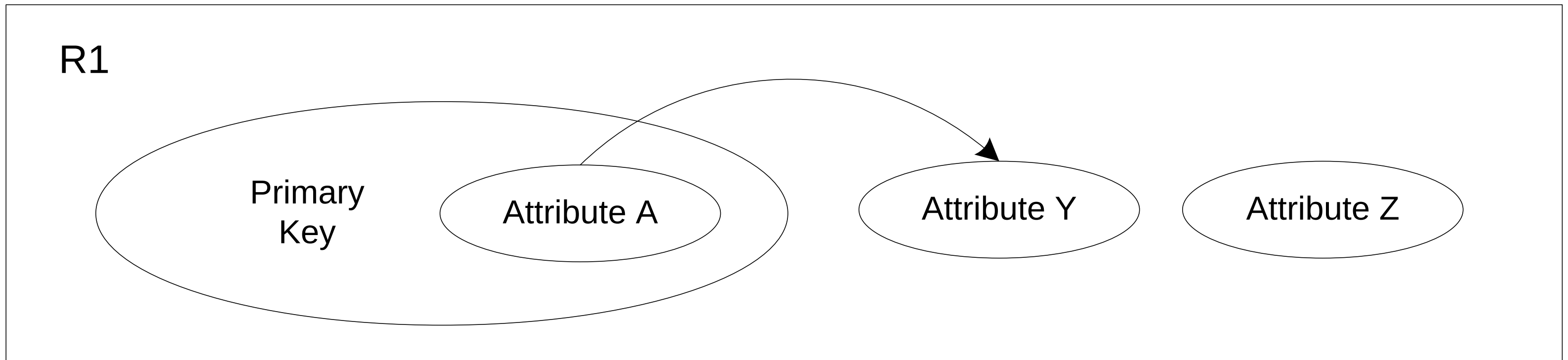
2nd Normal Form (2NF)

Second Normal Form (2NF) (no partial dependencies):

A relation schema is in 2nd normal form if

1 – it is in 1NF and

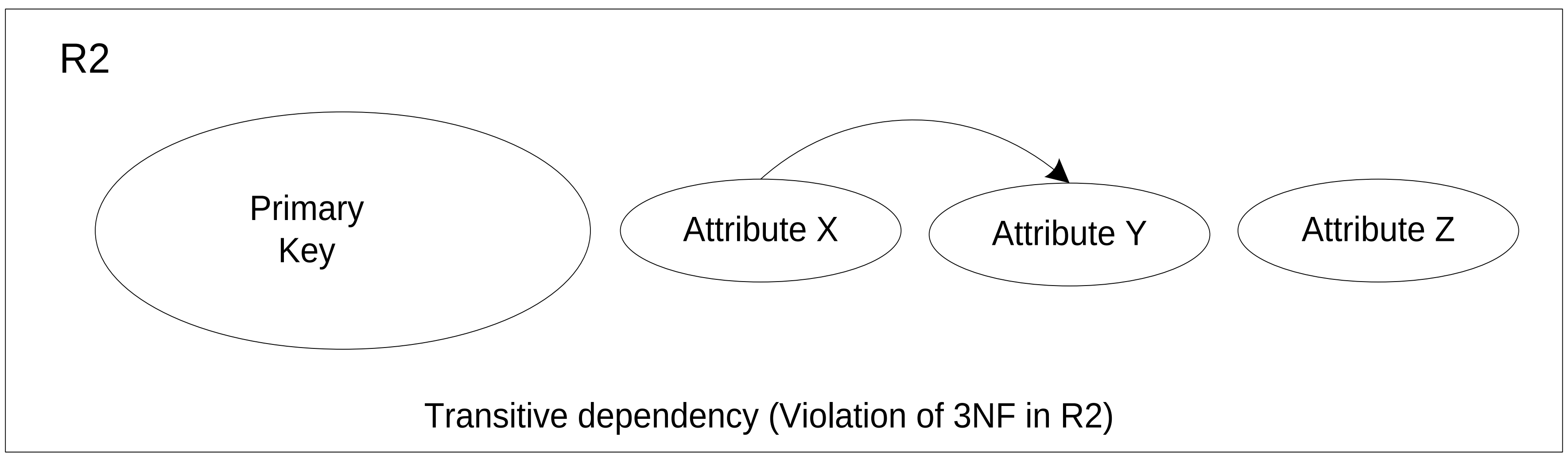
2 – every attribute is ***fully*** functionally dependent on the primary key of the relation. No partial dependency.

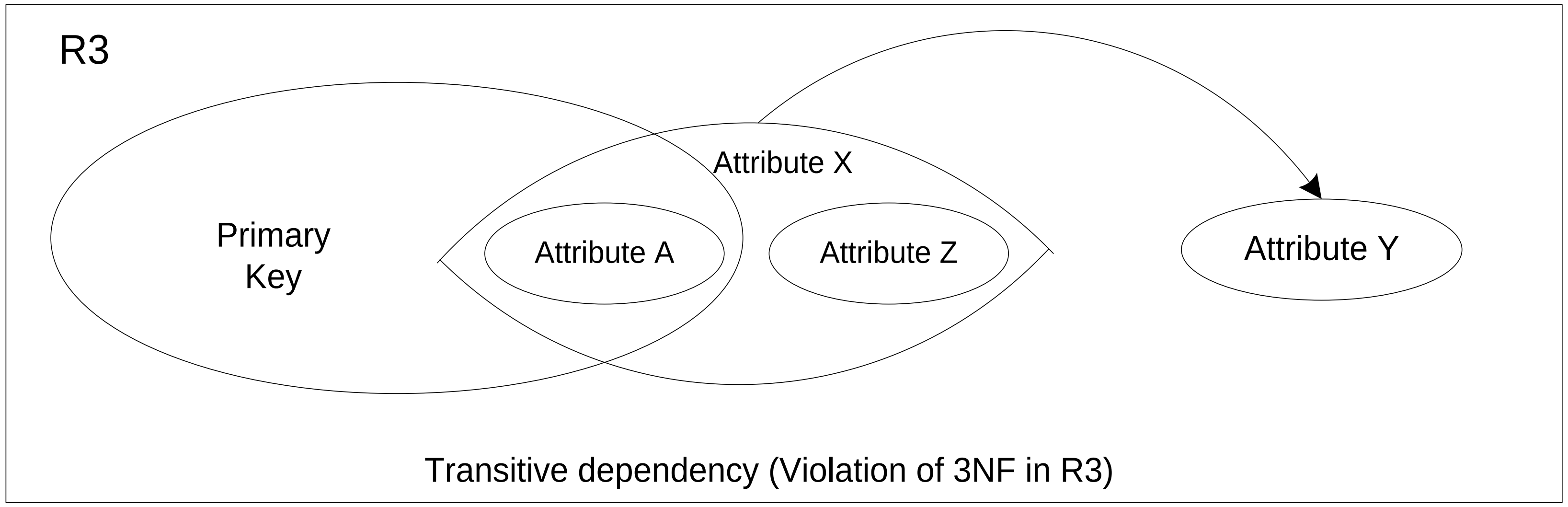


Partial dependency (Violation of 2NF in R1)

3rd Normal Form (3NF)

Third Normal Form (3NF) (no transitive dependencies):
A relation schema is in 3rd normal form if no non-prime attribute is functionally dependent on another non-prime attribute





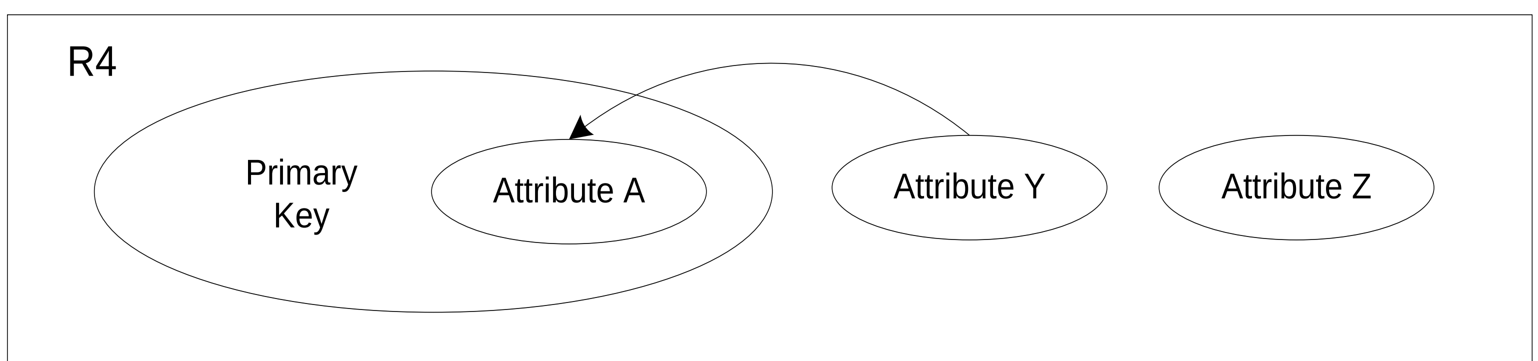
Boyce Codd Normal Form (BCNF)

Boyce-Codd Normal Form (BCNF):

A relation schema is in BCNF if

1 – it is in 3NF

2 – all FD determinants are subset of the primary key



Transitive dependency (Violation of BCNF in R4)

[Note: R4 is in 3NF]

Normalization Sales Pitch

Any non-key attribute in a relation schema must be

- about the key
- the whole key
- and nothing but the key

So help me Codd