Good and Bad Schemas

**Functional Dependencies**

**Even more normal forms!**

# Looking ahead…

Week 6 (now)

- Relational Algebra HW was due
- Flask+SQL Lab was due
- Lecture: Finishing Normal forms
- Lab: Using Amazon Web Services to run a database in the cloud

Week 7

- SQL Query HW due Wednesday (individual)
- Normalization HW due Wednesday (can discuss with others)
- Lecture: Project Overview, Exam Review
- Lab: Shopping Cart Project

NO LATE Submissions

Week 8

- **Monday 3/1: Midterm exam on Blackboard during class**
  - If you need to arrange a different time to take the exam (eg time zone issue, contact me by Wednesday 2/24

# Normal Forms 1-3

**1NF**: Attributes should be atomic and tables should have no repeating groups

- *Prevents messiness within a cell and repetition of rows*

**2NF**: There cannot be **X** → **A** where X is a partial candidate key for R

- Doesn't forbid non-prime to non-prime dependencies
- *Prevents repetition of cells across rows*

**3NF**: There cannot be **X** → **A** where X is not a full candidate key for R (unless A is a Key)

- Only allows dependencies on Keys
- *Prevents repetition of data within a row*

# Normal Form

Normal form reference:

- 2NF: Cannot have partial Key on left hand side (LHS)
- 3NF: Meet 2NF and LHS must be full Candidate Key or RHS must be a key

| ID | First name | Cid | Subj | Num | Grad |
|---|---|---|---|---|---|
| 1 | Sam | 570103 | SW | cs143 | B |
| 23 | Dan | 550103 | DB | cs178 | A |

**Functional Dependencies**

- ID → FirstName
- ID, Cid → Num, Grade
- Num → Subj

Can't be 2NF

violate 3NF

What normal form is this?

# Normal Form

Normal form reference:

- 2NF: Cannot have partial Key on left hand side (LHS)
- 3NF: Meet 2NF and LHS must be full Candidate Key or RHS must be a key

| ID | First name | Cid | Subj | Num | Grad |
|---|---|---|---|---|---|
| 1 | Sam | 570103 | SW | cs143 | B |
| 23 | Dan | 550103 | DB | cs178 | A |

**Functional Dependencies**

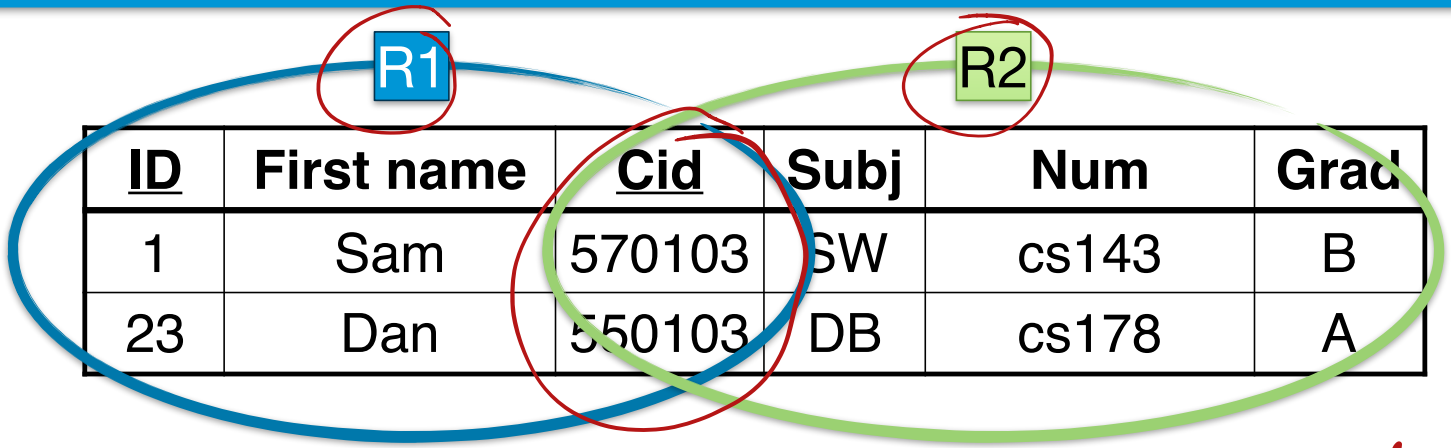
ID → FirstName
ID, Cid → Num, Grade
Num → Subj

partial key ID violates 2NF!

non-prime LHS would also violate 3NF!

Only meets 1NF

# How to Judge Decomposition?




| ID | First name | Cid | Subj | Num | Grad |
|---|---|---|---|---|---|
| 1 | Sam | 570103 | SW | cs143 | B |
| 23 | Dan | 550103 | DB | cs178 | A |

ID → FirstName
ID, Cid → Num, Grade
Num → Subj

F

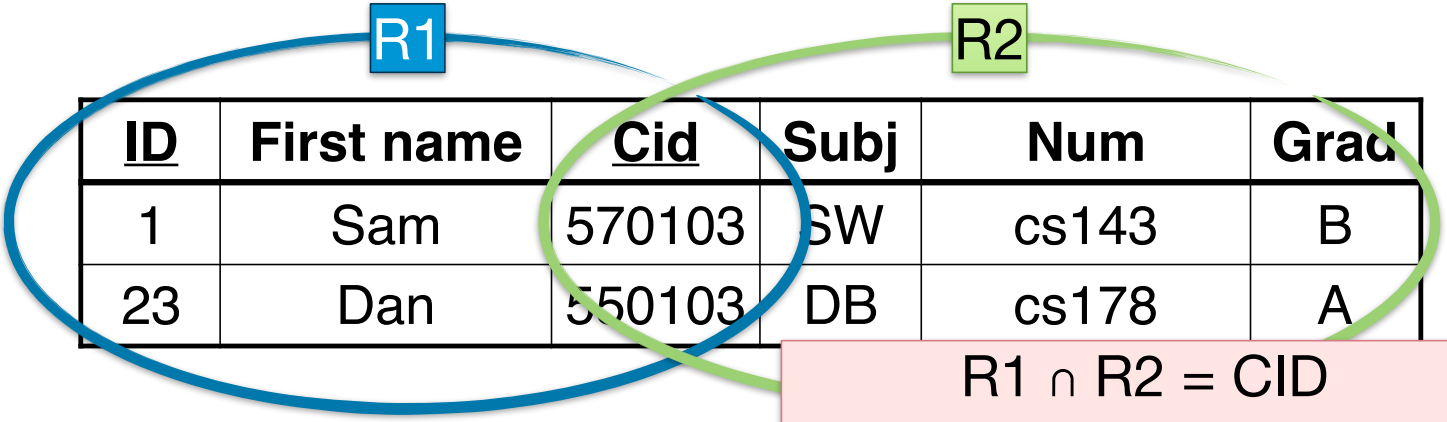
$R_1$ = ID, Name, Cid
$R_2$ Cid, Subj, Num, Grade
$R_1 \cap R_2$ = Cid

## Lossless Decomposition test:

- **R1**, **R2** is a lossless join decomposition of **R** with respect to **F** **iff** at least one of the following dependencies is in **F+**
- **(R1 ∩ R2) → R1 – R2**
- **(R1 ∩ R2) → R2 – R1**

• Cid → ID, Name
• Cid → Subj, Name, grade

# Lossless Decomposition




| ID | First name | Cid | Subj | Num | Grad |
|---|---|---|---|---|---|
| 1 | Sam | 570103 | SW | cs143 | B |
| 23 | Dan | 550103 | DB | cs178 | A |

ID → FirstName
ID, Cid → Num, Grade
Num → Subj

R1 ∩ R2 = CID

R1 - R2 = ID, First

R2 - R1 = Subj, Num, Grade

**Not lossless!**

Lossless Decomposition test:

- **R1**, **R2** is a lossless join decomposition of **R** with respect to **F** **iff** at least one of the following dependencies is in **F+**
- **(R1 ∩ R2) → R1 – R2**
- **(R1 ∩ R2) → R2 – R1**

# Dependency Preservation

We also must maintain dependences

After decomposition from **R** to **R1 … Rn**, the closure of FDs of all **R1…Rn** must be equivalent to that of **R**

**R1 = ID,** FirstName, CID
**R2 = CID,** Sub, Num, Grade

or

**R3 = ID,** FirstName
**R4 = ID, CID,** Sub, Num, Grade

ID → FirstName (1)
ID, Cid → Num, Grade (2)
Num → Subj (3)

# Dependency Preservation

We also must maintain dependences

After decomposition from **R** to **R1 … Rn**, the closure of FDs of all **R1…Rn** must be equivalent to that of **R**

**R1 = ID,** FirstName, CID
**R2 = CID,** Sub, Num, Grade

or

R1,R2 will lose the ID,CID -> Num, Grade FD

ID → FirstName

ID, Cid → Num, Grade

Num → Subj

**R3 = ID,** FirstName
**R4 = ID, CID,** Sub, Num, Grade

R3,R4 will maintain all FDs

# 3NF

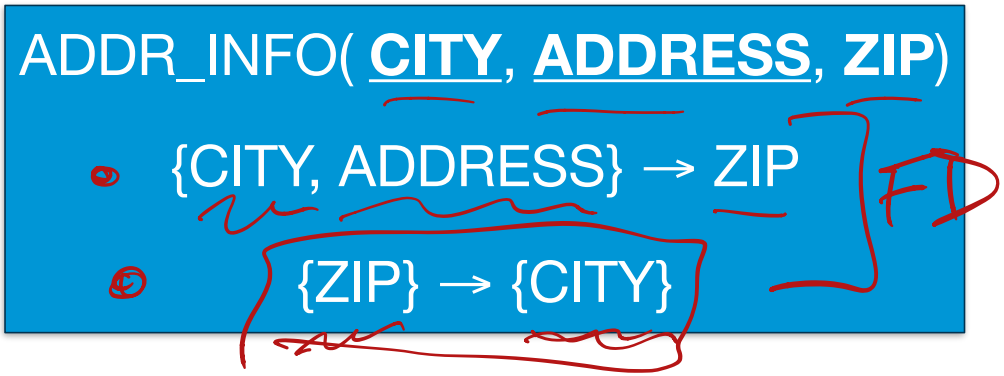
It is **always possible** to decompose a relation R into a set of relations R1…Rn which is **dependency preserving** and **lossless**

3NF is the baseline for acceptable DB normalization in practice!

but 3NF is not perfect…

# When does 3NF fail?

Suppose we want to store addresses:




Meets 3NF since LHS is a full Key **or** RHS is a Key

3NF: There cannot be **X → A** where X is not a full candidate key for R **(unless A is a Key)**

# When does 3NF fail?

ADDR_INFO( **CITY**, **ADDRESS**, **ZIP**)

{CITY, ADDRESS} → ZIP

{ZIP} → {CITY}

| City | Address | Zip |
|---|---|---|
| Washington | 800 22nd St NW | 20052 |
| Washington | 1600 Pennsylvania Avenue NW | 20050 |
| Philadelphia | Birch St | 20050 |
| | | |

3NF: There cannot be **X** → **A** where X is not a full candidate key for R **(unless A is a Key)**

# When does 3NF fail?

ADDR_INFO( **CITY**, **ADDRESS**, **ZIP**)

{CITY, ADDRESS} → ZIP

{ZIP} → {CITY}

| City | Address | Zip |
|---|---|---|
| Washington | 800 22nd St NW | 20052 |
| Washington | 1600 Pennsylvania Avenue NW | 20050 |
| Philadelphia | 101 South Street | 20050 |
| | | |

3NF does not prevent insertion/update of tuples which violate our FDs!

# 3NF vs BCNF

Third Normal Form (3NF): For every $\mathbf{X} \rightarrow \mathbf{A}$ that holds over relationship schema **R**,

1. either $\mathbf{A} \in \mathbf{X}$ (it is trivial), or
2. **X** is a superkey for **R**, or
3. **A** is a member of some key for **R**

Option 3 can result in update anomalies!

**Boise-Codd Normal Form** (BCNF) resolves this issue:

For every $\mathbf{X} \rightarrow \mathbf{A}$ that holds over relationship schema **R**,

1. either $\mathbf{A} \in \mathbf{X}$ (it is trivial), or
2. **X** is a superkey for **R**

# BCNF

BCNF is stricter than 3NF

- If a relation is in BCNF, it is also in 3NF;
- if it is not in 3NF, it is not in BCNF

Note:

- There are polynomial time algorithms **guaranteed to provide a lossless, dependency preserving decomposition** into 3NF
- **but** a **dependency preserving** decomposition into BCNF **may not exist**, and no polynomial time algorithm for **lossless decomposition** is known.

# Normalization Summary

**Functional Dependencies**: Capture the dependencies between attributes

**Normalization**: Provides a schema that ensures functional dependencies will be kept consistent, without losing data

**Normal Forms**: Try to achieve BCNF, but 3NF is OK in some cases (1NF/2NF -> bad design!)

To the cloud!

# *Why* use the cloud?

- Pay-as-you go
- Expand quickly on demand
- Don't need to worry about (many) IT issues
- Cheap!

Amazon AWS
Google CP
MS Azure

**... but is the cloud perfect?**

[spoiler alert] no.

# Infrastructure as a Service (IaaS)

Infrastructure clouds rent **raw servers**

- Connect to server remotely
- Configure OS and install whatever applications you want

Great flexibility for cloud user

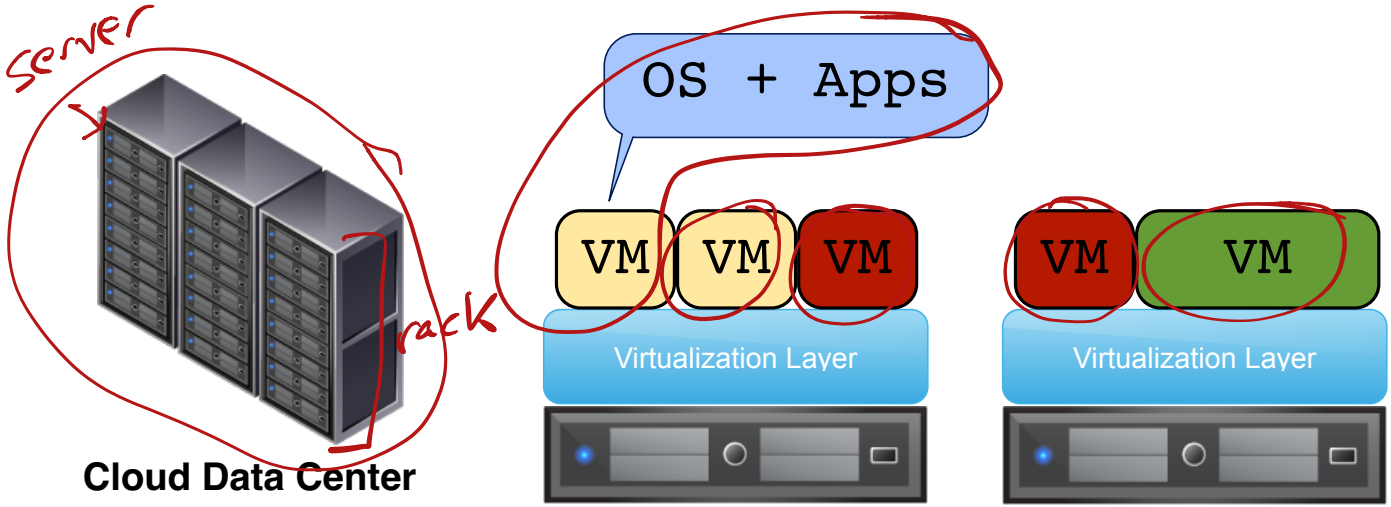
Less management handled by cloud operator

**Your own computer or disk on demand!**

# Virtual Machines

Virtualization is used to **split up** a physical server

- Allows multiple customers to share one machine
- Simplifies management since VMs are not strictly tied to HW
- Provides isolation between cloud users



# Amazon EC2

- Infrastructure as a Service Cloud (IaaS)

Economy of scale

- Can rent server and storage resources

| | Description | Cost |
|---|---|---|
| **t3.Micro** | 1GB RAM, up to 1 core, no storage | $0.01 / hour |
| **t3.Large** | 8GB RAM, ~2 cores, no storage | $0.08 / hour |
| **c5.18xlarge** | 144GB RAM, 72 cores, no storage | $3.06 / hour |

| | | |
|---|---|---|
| **EBS** | Network attached storage | $0.10 / GB per month |