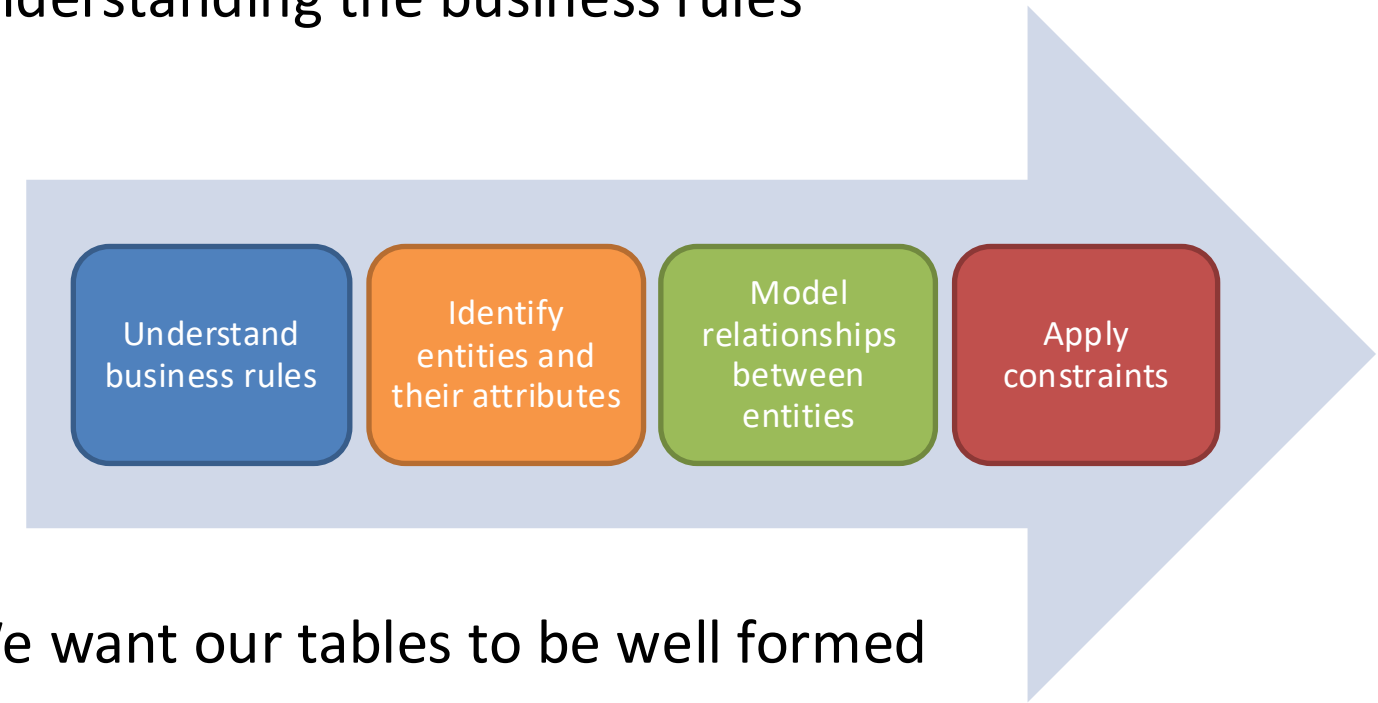
CS 61:
Database Systems

Normalization

Adapted from Coronel and Morris unless otherwise noted
Augmented by Claude Opus

Objective: create well-formed relations

- Tables are the building blocks of a relational database
- Previously we created tables for entities identified after understanding the business rules

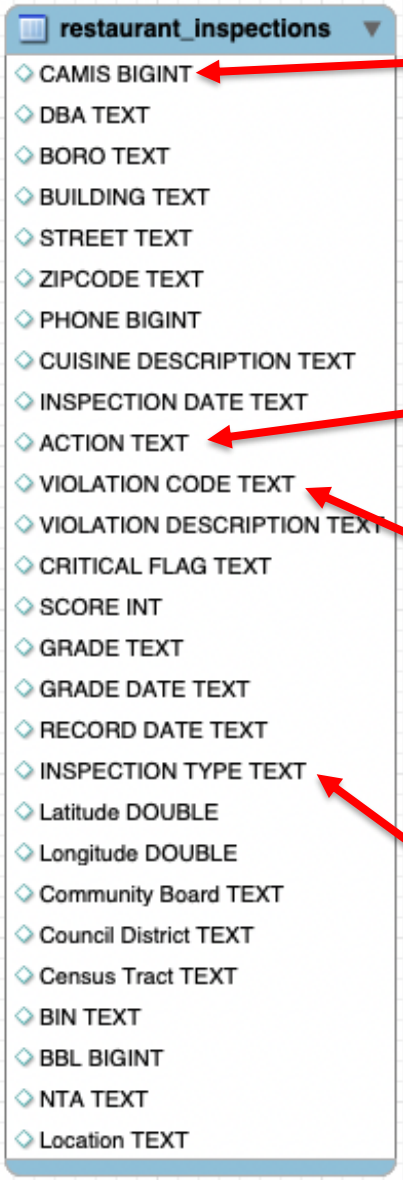


- We want our tables to be well formed
- Question, how do we know if our tables are well formed?
- It turns out a few relatively simple rules can help us

Agenda

1. Theoretical normalization
2. Practical normalization

Recall where we started with health inspection data



Each restaurant is identified by a unique ID (CAMIS)

Each restaurant can be inspected many times

The inspector may take different actions (close restaurant, re-open, etc) stored in Action

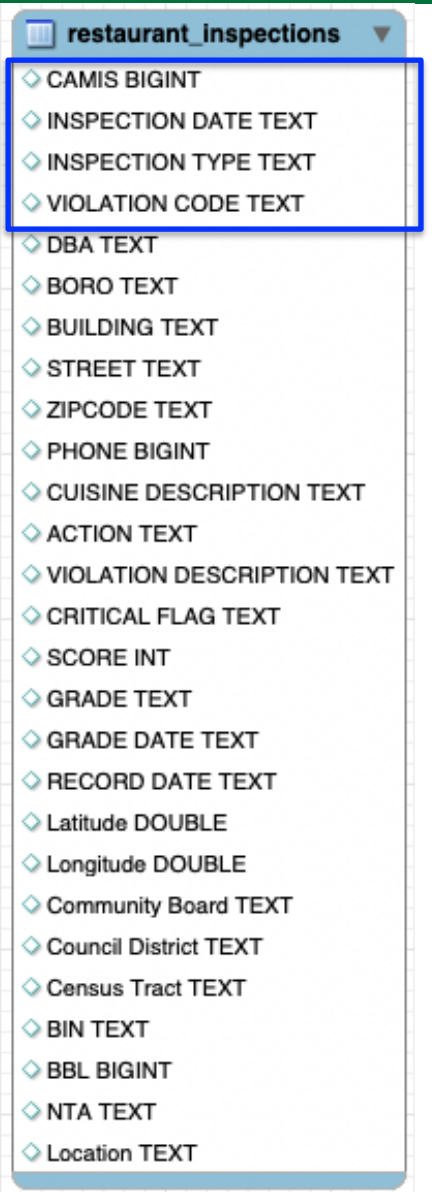
Each inspection may result in multiple violations, where each violation is identified by a violation code

So, one inspection may result in many rows in the table

Also: violation description and critical flag go with violation code

Each inspection for is a reason (inspection type)
Can have multiple types of inspections on same day

Recall where we started with health inspection data



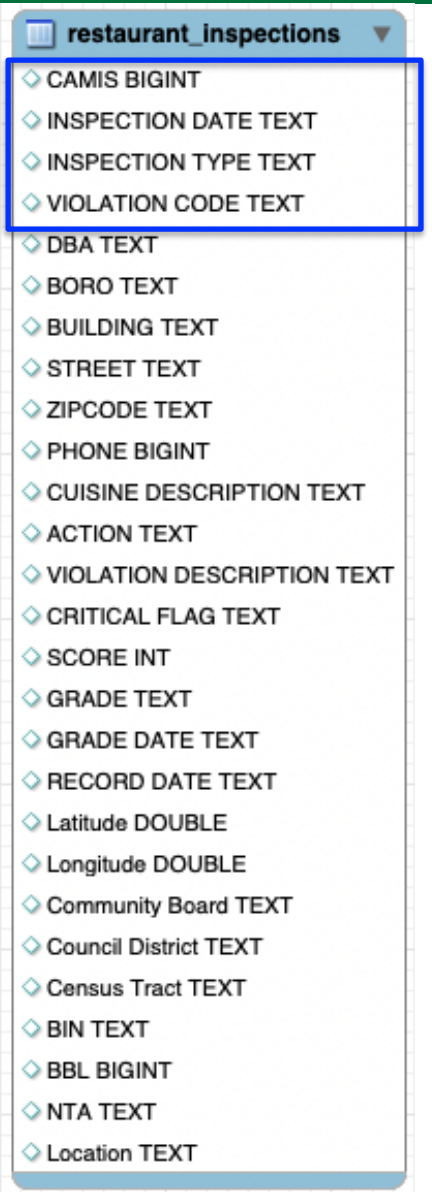
We can uniquely identify each row by

- **CAMIS**
- **Inspection Date**
- **Inspection Type**
- **Violation Code**

This 4-tuple can be a Primary Key on the table

(We *can* use this tuple as a Primary Key, but we might not want to!)

Recall where we started with health inspection data



What sort of anomalies can occur?

- **Insert anomalies (can't have a restaurant that hasn't been inspected)**
- **Update/consistency anomalies (restaurant name stored for each inspection, must update all of them if the name changes)**
- **Delete anomalies (if a restaurant was inspected one time and we delete it, we lose track of the restaurant)**

Normalization is about correcting table structure to minimize data redundancy

Normalization

- Works in a series of stages called normal forms
- First normal form (1NF) through third normal form (3NF) or higher
- High forms tend to split relations into multiple relations, each with fewer attributes
- Generally, the higher the form, the more joins are required to produce data
  - More resources required by the database to respond to requests
  - Slower performance
- Occasionally we will denormalize tables
  - Denormalization may result in redundant/dependent data
  - Particularly common in reporting/analysis databases
  - Deciding when to denormalize is part of the “art” of good database design

Key review

Key type	Definition
Superkey	An attribute or combination of attributes that uniquely identifies each row in a table
Candidate key	A minimal (irreducible) superkey; a superkey that does not contain a subset of attributes that is itself a superkey
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row; cannot contain null entries
Foreign key	An attribute or combination of attributes in one table whose values must either match the primary key in another table or be null
Composite key	A key comprised of multiple attributes (sometimes called a compound key)
Surrogate key	A system assigned key, often auto incrementing

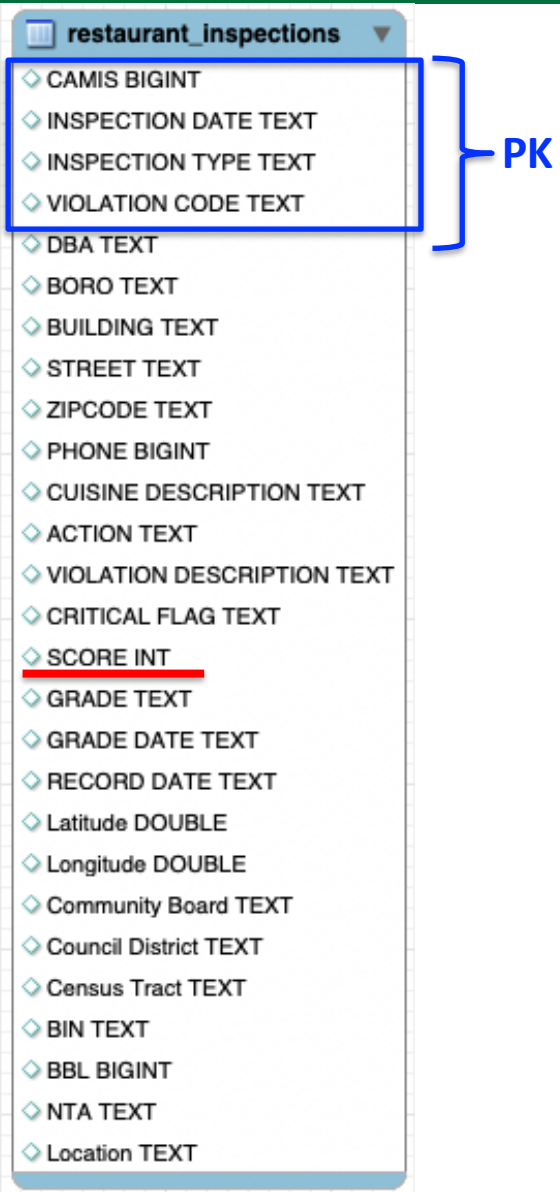
Functional dependence is a generalized notion of keys

Functional dependence (FD)

- One or more attributes determine the the value of one or more other attributes in a relation
- This role of a key: to determine the value of other attributes
- Written A → B
  - A is called the **determinant**
  - B is called the **dependent** (value identified by A)
  - Here A is the (possibly composite) key and B is a collection of attributes that can be looked up given key A
  - We say “A determines B”

Can look up B, if given A

Full functional dependence: all Primary Key attributes are needed to identify attribute



Full functional dependence

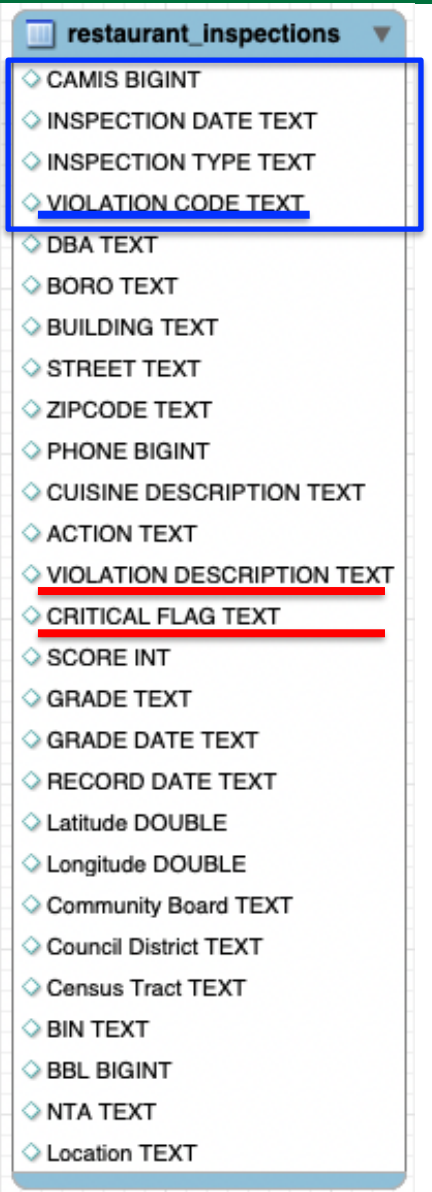
An attribute is functionally dependent on a composite key but not any subset of the key (e.g., all attributes in key are required)

Ex: CAMIS, Inspection Date, Inspection Type, ViolationCode → Score

All four attributes are required to uniquely identify the score

Full dependence is what we want!
(full dependence is good)

Partial dependence: only part of the Primary Key is needed to identify attribute



Partial dependence

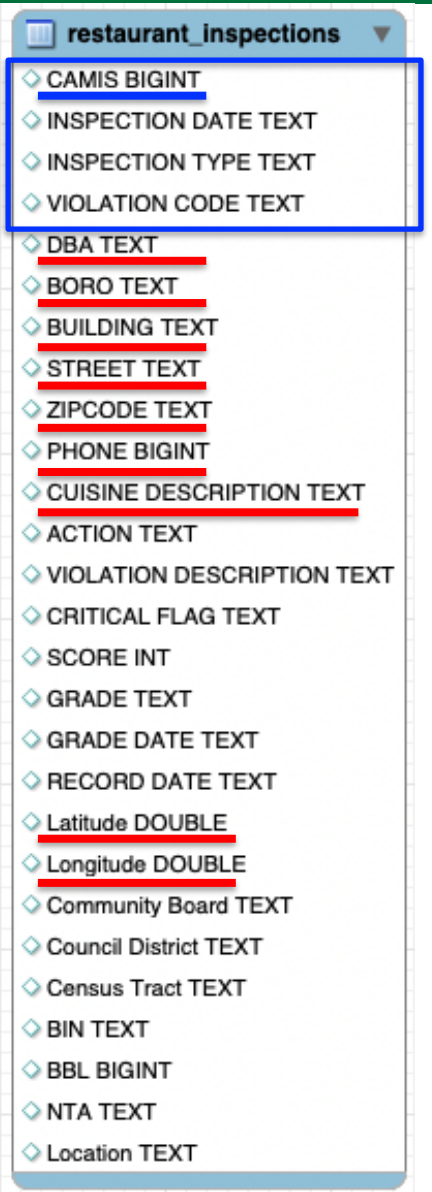
An attribute is dependent on only part of the key

Ex: ViolationCode → ViolationDescription, Critical Flag

Violation Description and Critical Flag only depend on ViolationCode (part of the composite primary key)

Partial dependencies are (normally) straight forward, easy to identify

Partial dependence: only part of the Primary Key is needed to identify attribute



Partial dependence

An attribute is dependent on only part of the key

Ex: ViolationCode → ViolationDescription, Critical Flag

Violation Description and Critical Flag only depend on ViolationCode (part of the composite primary key)

Partial dependencies are (normally) straight forward, easy to identify

Others?

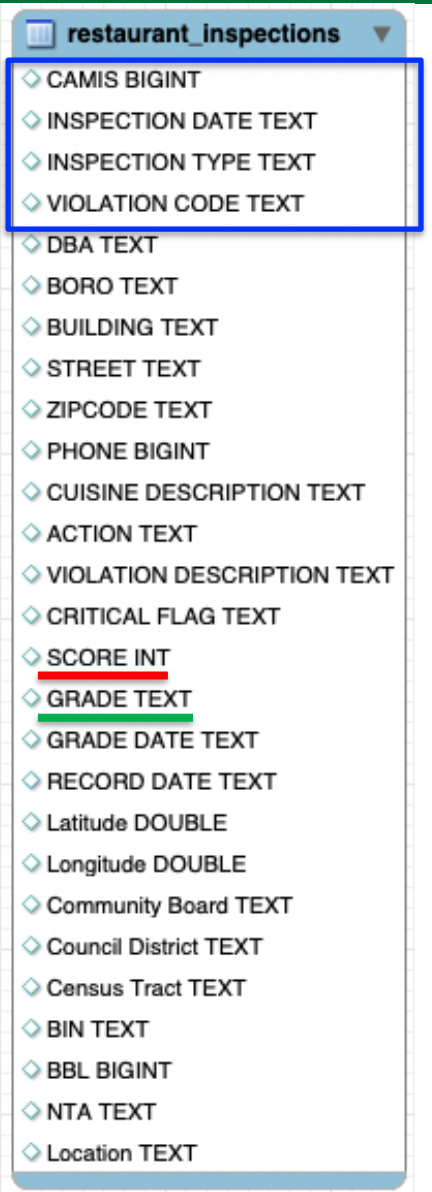
CAMIS → dba, boro, building, street, zipcode, phone, cuisine description, latitude, longitude

Partial dependence leads to storing data more than necessary, can look up dba, boro, … from CAMIS

Anomalies:

- **Insert**
- **Update**
- **Delete**

Transitive dependence: non-key attribute determines another attribute



Transitive dependence

If A→B and B→C, then A→C

An attribute is dependent on another attribute that is not part of the key

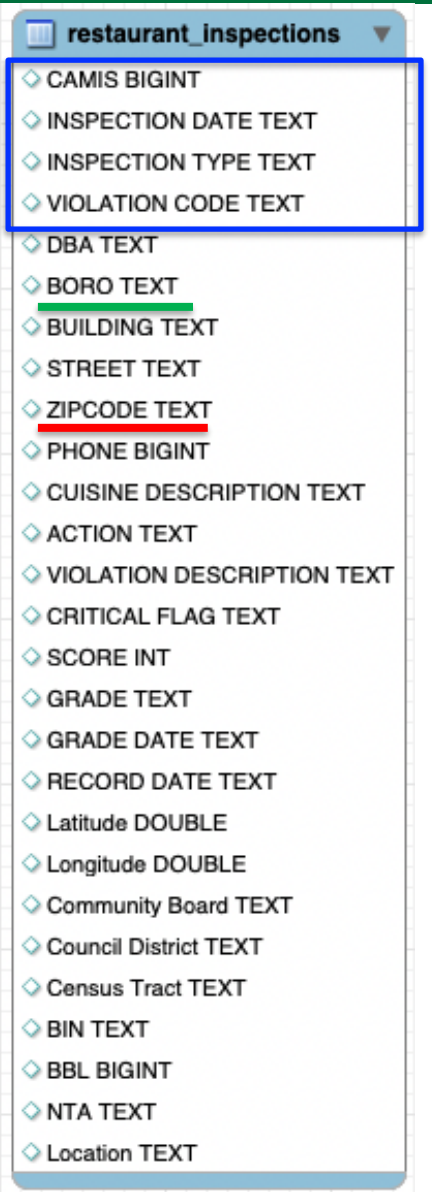
More difficult to identify among a set of data

Occurs when functional dependence exists among nonprime attributes

Ex: PK → Score → Grade

Assume grade:
A if Score < 10
B if Score < 20
C if Score < 30
…

Transitive dependence: non-key attribute determines another attribute



Transitive dependence

If A→B and B→C, then A→C

An attribute is dependent on another attribute that is not part of the key

More difficult to identify among a set of data

Occurs when functional dependence exists among nonprime attributes

Ex: PK → Score → Grade

PK → Zipcode → Boro

Transitive dependence leads to storing data more than necessary: can look up grade from score, or boro from zipcode

Anomalies:

Insert

Update

Delete

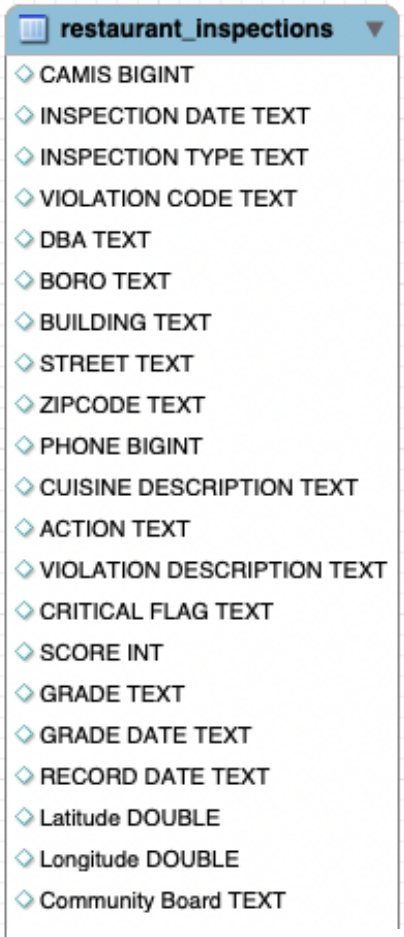
To combat issues: we will move from First to Third Normal Form

First (1NF), Second (2NF) and Third (3NF) normal form characteristics

First Normal Form (1NF)

1. Data in table format
2. No repeating groups
3. PK identified
4. All dependencies identified

Start with 1NF: table form, no repeating groups, PK and dependencies identified

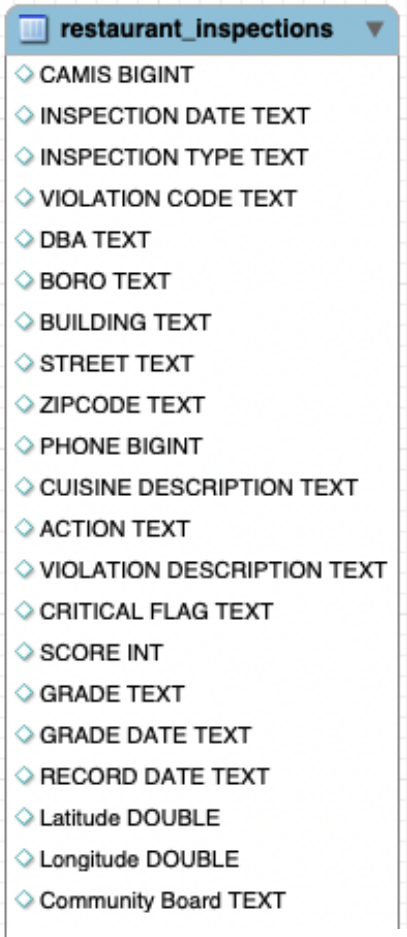


1: Put in table form

- Already done

RestaurantID	InspectionDa...	InspectionType	ViolationCode	ViolationDescription	ActionDecription	Score	CriticalFl
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04H, 04L, 04N, 06C, 06D, 08C, 10D, 10F	Raw, cooked or prepared food is adulterated, co…	Establishment Closed by DOHMH. Violations w…	21	Y
30075445	2023-02-03	Cycle Inspection / Reopening Inspection	02G, 10F	Non-food contact surface or equipment made of…	Establishment re-opened by DOHMH.	13	Y
30075445	2023-08-01	Cycle Inspection / Initial Inspection	04L, 05H, 08A	No approved written standard operating proced…	Violations were cited in the following area(s).	38	Y
30075445	2023-08-22	Cycle Inspection / Re-inspection	04L, 08A, 08C	Pesticide not properly labeled or used by unlice…	Violations were cited in the following area(s).	12	Y
30075445	2024-11-08	Cycle Inspection / Initial Inspection	06C, 08C, 10F	Pesticide not properly labeled or used by unlice…	Violations were cited in the following area(s).	10	Y
30191841	2023-04-23	Cycle Inspection / Initial Inspection	06C, 06E	Sanitized equipment or utensil, including in-use…	Violations were cited in the following area(s).	10	Y
30191841	2024-11-20	Cycle Inspection / Initial Inspection	04L, 06C, 08A, 08C, 09B, 10F	Thawing procedure improper.	Violations were cited in the following area(s).	24	Y
30191841	2025-02-20	Cycle Inspection / Re-inspection	06C, 06D	Food, supplies, or equipment not protected from…	Violations were cited in the following area(s).	10	Y

Start with 1NF: table form, no repeating groups, PK and dependencies identified



1: Put in table form

2: Eliminate repeating groups

- Multiple entries ViolationCode and ViolationDescription attributes because each inspection may result in multiple violations
- Remove repeating entries by making each violation its own row

Another variant of repeating groups: Column for Violation1, Violation2, … Violation n

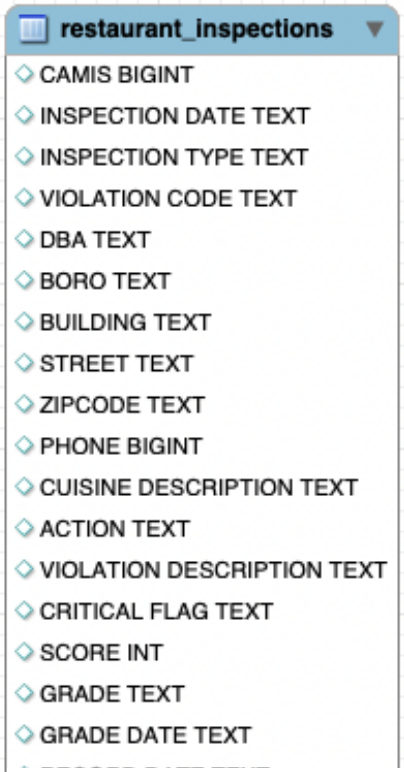
Repeating groups

Make one column (ViolationCode) and make duplicate rows but each row has a unique ViolationCode

Now ViolationCode is atomic

RestaurantID	InspectionDa...	InspectionType	ViolationCode	ViolationDescription	ActionDecription	Score	CriticalFl
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04H, 04L, 04N, 06C, 06D, 08C, 10D, 10F	Raw, cooked or prepared food is adulterated, co…	Establishment Closed by DOHMH. Violations w…	21	Y
30075445	2023-02-03	Cycle Inspection / Reopening Inspection	02G, 10F	Non-food contact surface or equipment made of…	Establishment re-opened by DOHMH.	13	Y
30075445	2023-08-01	Cycle Inspection / Initial Inspection	04L, 05H, 08A	No approved written standard operating proced…	Violations were cited in the following area(s).	38	Y
30075445	2023-08-22	Cycle Inspection / Re-inspection	04L, 08A, 08C	Pesticide not properly labeled or used by unlice…	Violations were cited in the following area(s).	12	Y
30075445	2024-11-08	Cycle Inspection / Initial Inspection	06C, 08C, 10F	Pesticide not properly labeled or used by unlice…	Violations were cited in the following area(s).	10	Y
30191841	2023-04-23	Cycle Inspection / Initial Inspection	06C, 06E	Sanitized equipment or utensil, including in-use…	Violations were cited in the following area(s).	10	Y
30191841	2024-11-20	Cycle Inspection / Initial Inspection	04L, 06C, 08A, 08C, 09B, 10F	Thawing procedure improper.	Violations were cited in the following area(s).	24	Y
30191841	2025-02-20	Cycle Inspection / Re-inspection	06C, 06D	Food, supplies, or equipment not protected from…	Violations were cited in the following area(s).	10	Y

Start with 1NF: table form, no repeating groups, PK and dependencies identified



1: Put in table form

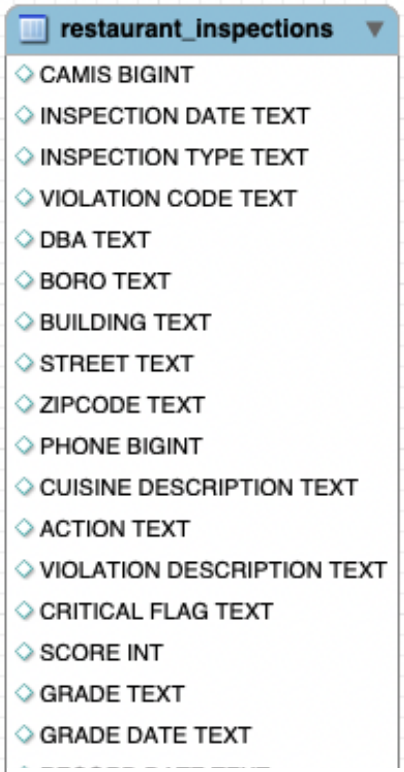
2: Eliminate repeating groups

- Multiple entries ViolationCode and ViolationDescription attributes because each inspection may result in multiple violations
- Remove repeating entries by making each violation its own row

Now each violation on its own row

RestaurantID	InspectionDa...	InspectionType	ViolationCode	ViolationDescription	ActionDescription	Score	CriticalFlag	Grade
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04H	Raw, cooked or prepared food is adulterated, co...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04L	Evidence of mice or live mice in establishment's...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04N	Filth flies or food/refuse/sewage associated with...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	06C	Food, supplies, and equipment not protected fro...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	06D	Food contact surface not properly washed, rinse...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	08C	Pesticide not properly labeled or used by unlice...	Establishment Closed by DOHMH. Violations w...	21	N	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	10D	Mechanical or natural ventilation not provided, i...	Establishment Closed by DOHMH. Violations w...	21	N	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	10F	Non-food contact surface or equipment made of...	Establishment Closed by DOHMH. Violations w...	21	N	
30075445	2023-02-03	Cycle Inspection / Reopening Inspection	02G	Cold TCS food item held above 41 °F; smoked...	Establishment re-opened by DOHMH.	13	Y	P
30075445	2023-02-03	Cycle Inspection / Reopening Inspection	10F	Non-food contact surface or equipment made of...	Establishment re-opened by DOHMH.	13	N	P
30075445	2023-08-01	Cycle Inspection / Initial Inspection	04L	Evidence of mice or live mice in establishment's...	Violations were cited in the following area(s).	38	Y	
30075445	2023-08-01	Cycle Inspection / Initial Inspection	05H	No approved written standard operating proced...	Violations were cited in the following area(s).	38	Y	
30075445	2023-08-01	Cycle Inspection / Initial Inspection	08A	Establishment is not free of harborage or conditi...	Violations were cited in the following area(s).	38	N	
30075445	2023-08-22	Cycle Inspection / Re-inspection	04L	Evidence of mice or live mice in establishment's...	Violations were cited in the following area(s).	12	Y	A

Start with 1NF: table form, no repeating groups, PK and dependencies identified



1: Put in table form

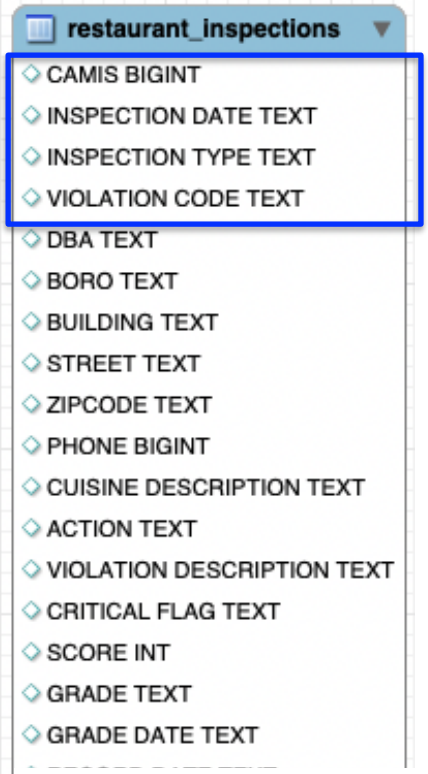
2: Eliminate repeating groups

- Multiple entries ViolationCode and ViolationDescription attributes because each inspection may result in multiple violations
- Remove repeating entries by making each violation its own row

RestaurantID	InspectionDa...	InspectionType	ViolationCode	ViolationDescription	ActionDescription	Score	CriticalFlag	Grade
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04H	Raw, cooked or prepared food is adulterated, co...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04L	Evidence of mice or live mice in establishment's...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04N	...ith...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	06C	...ro...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	06D	...se...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	08C	...e...	Establishment Closed by DOHMH. Violations w...	21	N	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	10D	...i...	Establishment Closed by DOHMH. Violations w...	21	N	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	10F	Non-food contact surface or equipment made of...	Establishment Closed by DOHMH. Violations w...	21	N	
30075445	2023-02-03	Cycle Inspection / Reopening Inspection	02G	Cold TCS food item held above 41 °F; smoked...	Establishment re-opened by DOHMH.	13	Y	P
30075445	2023-02-03	Cycle Inspection / Reopening Inspection	10F	Non-food contact surface or equipment made of...	Establishment re-opened by DOHMH.	13	N	P
30075445	2023-08-01	Cycle Inspection / Initial Inspection	04L	Evidence of mice or live mice in establishment's...	Violations were cited in the following area(s).	38	Y	
30075445	2023-08-01	Cycle Inspection / Initial Inspection	05H	No approved written standard operating proced...	Violations were cited in the following area(s).	38	Y	
30075445	2023-08-01	Cycle Inspection / Initial Inspection	08A	Establishment is not free of harborage or conditi...	Violations were cited in the following area(s).	38	N	
30075445	2023-08-22	Cycle Inspection / Re-inspection	04L	Evidence of mice or live mice in establishment's...	Violations were cited in the following area(s).	12	Y	A

These rows are for one inspection

Start with 1NF: table form, no repeating groups, PK and dependencies identified



1: Put in table form

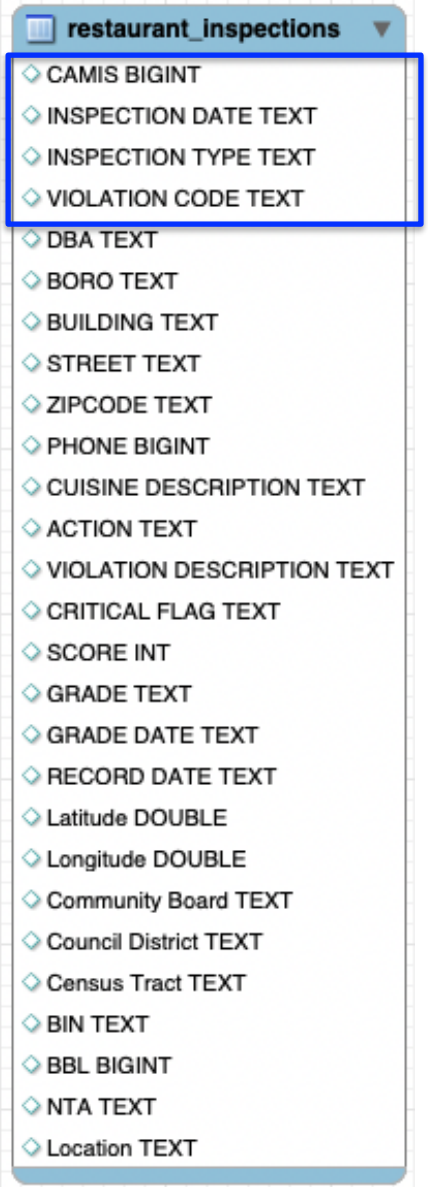
2: Eliminate repeating groups

3: Identify primary key

- Primary key: CAMIS, InspectionDate, InspectionType, ViolationCode
- Consider surrogate key
  - Four attributes becomes unwieldy
  - Surrogate key is a key assigned by the system (auto increment or UUID)

RestaurantID	InspectionDa...	InspectionType	ViolationCode	ViolationDescription	ActionDescription	Score	CriticalFlag	Grade
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04H	Raw, cooked or prepared food is adulterated, co...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04L	Evidence of mice or live mice in establishment's...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	04N	Filth flies or food/refuse/sewage associated with...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	06C	Food, supplies, and equipment not protected fro...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	06D	Food contact surface not properly washed, rinse...	Establishment Closed by DOHMH. Violations w...	21	Y	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	08C	Pesticide not properly labeled or used by unlice...	Establishment Closed by DOHMH. Violations w...	21	N	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	10D	Mechanical or natural ventilation not provided, i...	Establishment Closed by DOHMH. Violations w...	21	N	
30075445	2023-01-31	Cycle Inspection / Initial Inspection	10F	Non-food contact surface or equipment made of...	Establishment Closed by DOHMH. Violations w...	21	N	
30075445	2023-02-03	Cycle Inspection / Reopening Inspection	02G	Cold TCS food item held above 41 °F; smoked...	Establishment re-opened by DOHMH.	13	Y	P
30075445	2023-02-03	Cycle Inspection / Reopening Inspection	10F	Non-food contact surface or equipment made of...	Establishment re-opened by DOHMH.	13	N	P
30075445	2023-08-01	Cycle Inspection / Initial Inspection	04L	Evidence of mice or live mice in establishment's...	Violations were cited in the following area(s).	38	Y	
30075445	2023-08-01	Cycle Inspection / Initial Inspection	05H	No approved written standard operating proced...	Violations were cited in the following area(s).	38	Y	
30075445	2023-08-01	Cycle Inspection / Initial Inspection	08A	Establishment is not free of harborage or conditi...	Violations were cited in the following area(s).	38	N	
30075445	2023-08-22	Cycle Inspection / Re-inspection	04L	Evidence of mice or live mice in establishment's...	Violations were cited in the following area(s).	12	Y	A

Start with 1NF: table form, no repeating groups, PK and dependencies identified

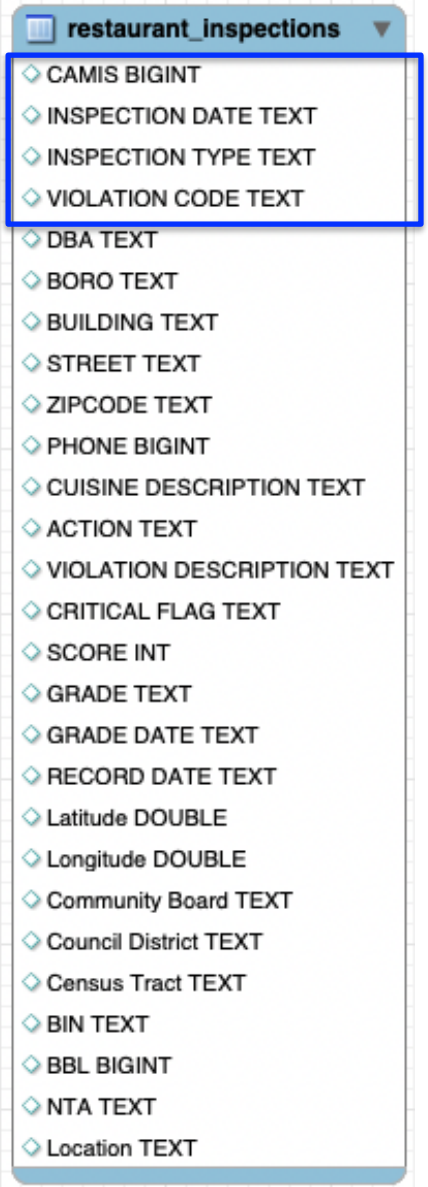


1: Put in table form
2: Eliminate repeating groups
3: Identify primary key
4: All dependencies identified

Partial

- **CAMIS** -> dba, boro, building, street, zipcode, phone, cuisine description, latitude, longitude (dropping attributes after longitude for simplicity)
- **Inspection Date, Inspection Type** -> action, violation description, critical flag, score, grade, grade date, record date

Start with 1NF: table form, no repeating groups, PK and dependencies identified



1: Put in table form
2: Eliminate repeating groups
3: Identify primary key
4: All dependencies identified

Partial

- **CAMIS** -> dba, boro, building, street, zipcode, phone, cuisine description, latitude, longitude (dropping attributes after longitude for simplicity)
- **Inspection Date, Inspection Type** -> action, violation description, critical flag, score, grade, grade date, record date

Transitive

- **Score** -> Grade
- **Zipcode** -> boro

Now in 1NF, move on to 2NF

Move to 2NF: remove partial dependencies

First (1NF), Second (2NF) and Third (3NF) normal form characteristics

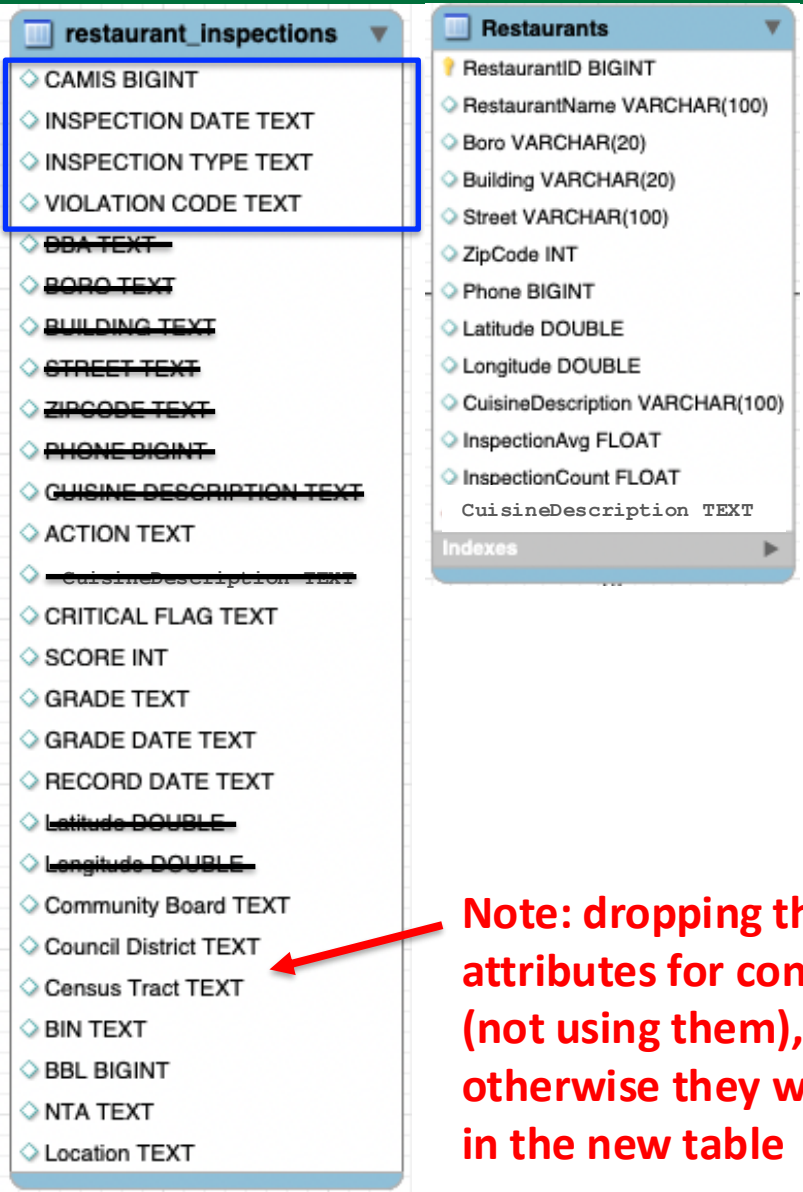
First Normal Form (1NF)

1. Data in table format
2. No repeating groups
3. PK identified
4. All dependencies identified

Second Normal Form (2NF)

- 1NF plus
- No partial dependencies

2NF: Remove partial dependencies by making new tables



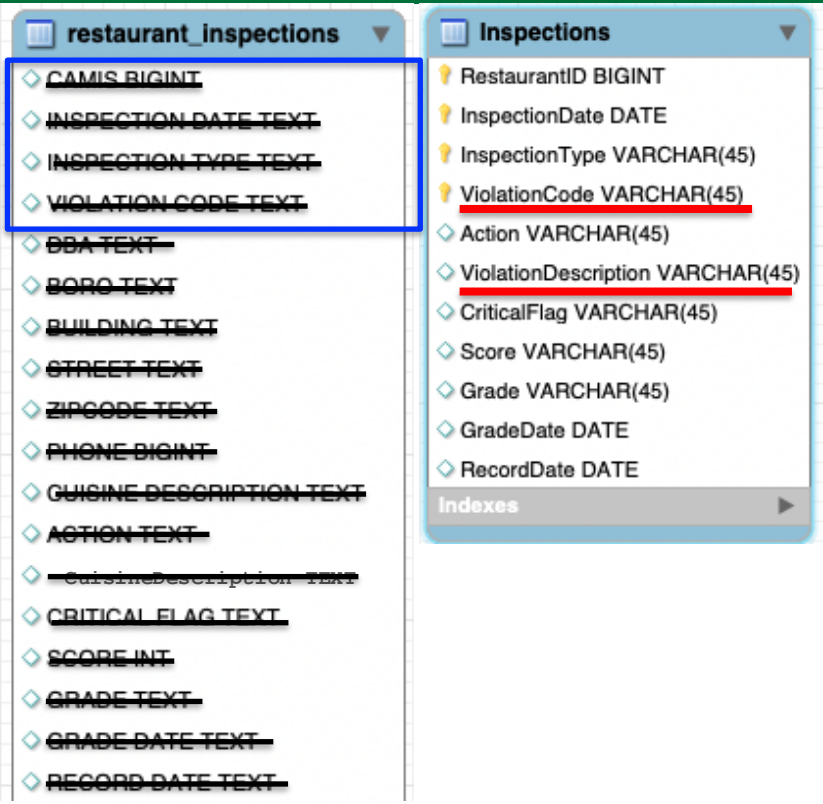
Make new tables to eliminate partial dependencies

- Partial dependency is when an attribute is dependent on only part of a composite key
- Two partial dependencies: **CAMIS** and **Inspections**
- **CAMIS**: Make Restaurants table with dependent attributes
  - Rename CAMIS to RestaurantID for convenience
  - Use RestaurantID as Primary Key
  - This table has a single attribute (surrogate) Primary Key, so there cannot have partial dependencies in it
  - Automatically 2NF

Note: dropping these attributes for convenience (not using them), otherwise they would go in the new table

Solve partial dependencies by creating a new table and move dependent attributes into the new table

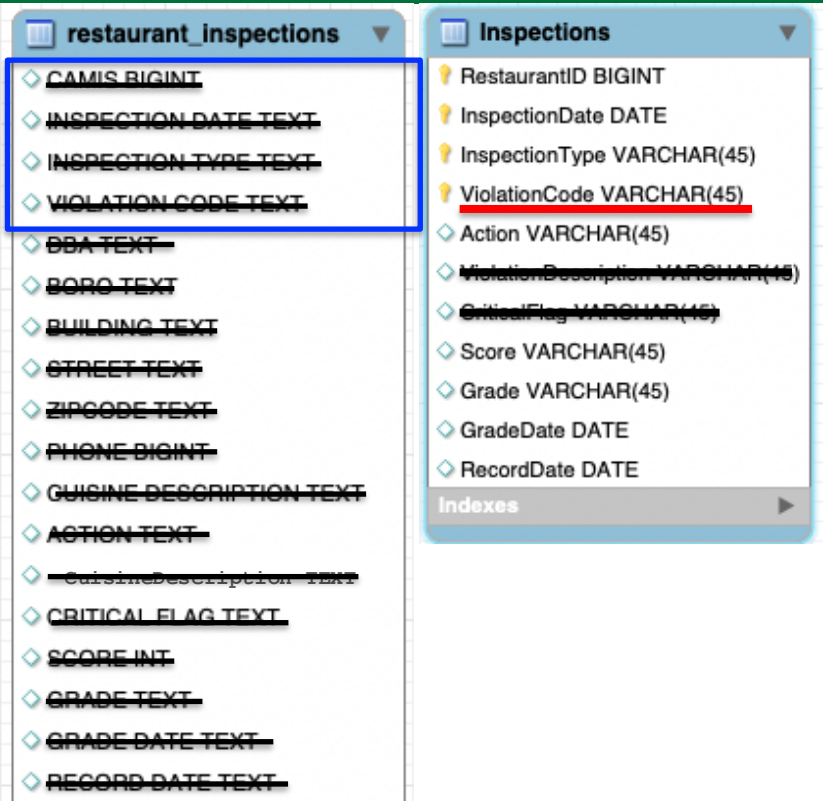
2NF: Remove partial dependencies by making new tables



Make new tables to eliminate partial dependencies

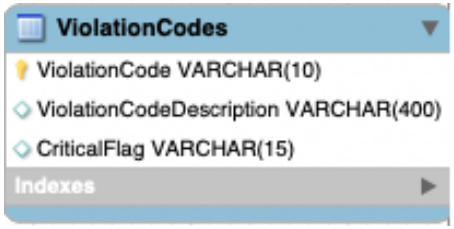
- **Inspections**: Make Inspections table with dependent attributes
  - Renaming most attributes (for style) and giving proper domain types
  - Using **RestaurantID, InspectionDate, InspectionType, ViolationCode** as Primary Key
  - Leaves us with partial dependencies
    - ViolationCode -> ViolationDescription, Critical Flag

2NF: Remove partial dependencies by making new tables



Make new tables to eliminate partial dependencies

- **Inspections**: Make Inspections table with dependent attributes
  - Renaming most attributes (for style) and giving proper domain types
  - Using **RestaurantID, InspectionDate, InspectionType, ViolationCode** as Primary Key
  - Leaves us with partial dependencies
    - ViolationCode -> ViolationDescription, Critical Flag
  - Make a new table for ViolationCodes with ViolationCodeDescription and Critical Flag



May also add a surrogate key to Inspections

We will examine one table at a time, moving from First to Third Normal Form

First (1NF), Second (2NF) and Third (3NF) normal form characteristics

First Normal Form (1NF)

1. Data in table format
2. No repeating groups
3. PK identified
4. All dependencies identified

Second Normal Form (2NF)

- 1NF plus
- No partial dependencies

Third Normal Form (3NF)

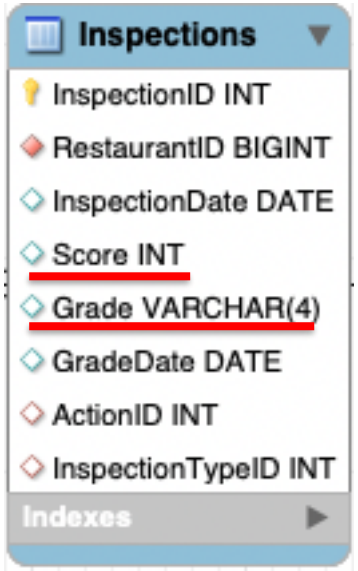
- 2NF plus
- No transitive dependencies

3NF: remove transitive from all tables

Make new tables to eliminate transitive dependencies

- Transitive dependency: If A→B and B→C, then A→C
- Identify by looking for dependencies on nonprime attributes
- RestaurantID -> Zipcode -> Boro
  - Could make look up table for boro based on Zipcode
  - Here I choose not to (I'm leaving the database somewhat denormalized as a result)
- Another design choice: make lookup table for Cuisine
  - Give Restaurants a foreign key into new Cuisine table
  - Now Italian/italian/ITALIAN consistency problem solved
  - Easier to rename Italian to Italian/Mediterranean if desired
  - Lower storage requirements
    - Only store one 4-byte integer for each restaurant
    - Rather than 2 bytes per character (Italian takes 14 bytes to store)
  - This is a design choice, not a normalization step

3NF: remove transitive from all tables

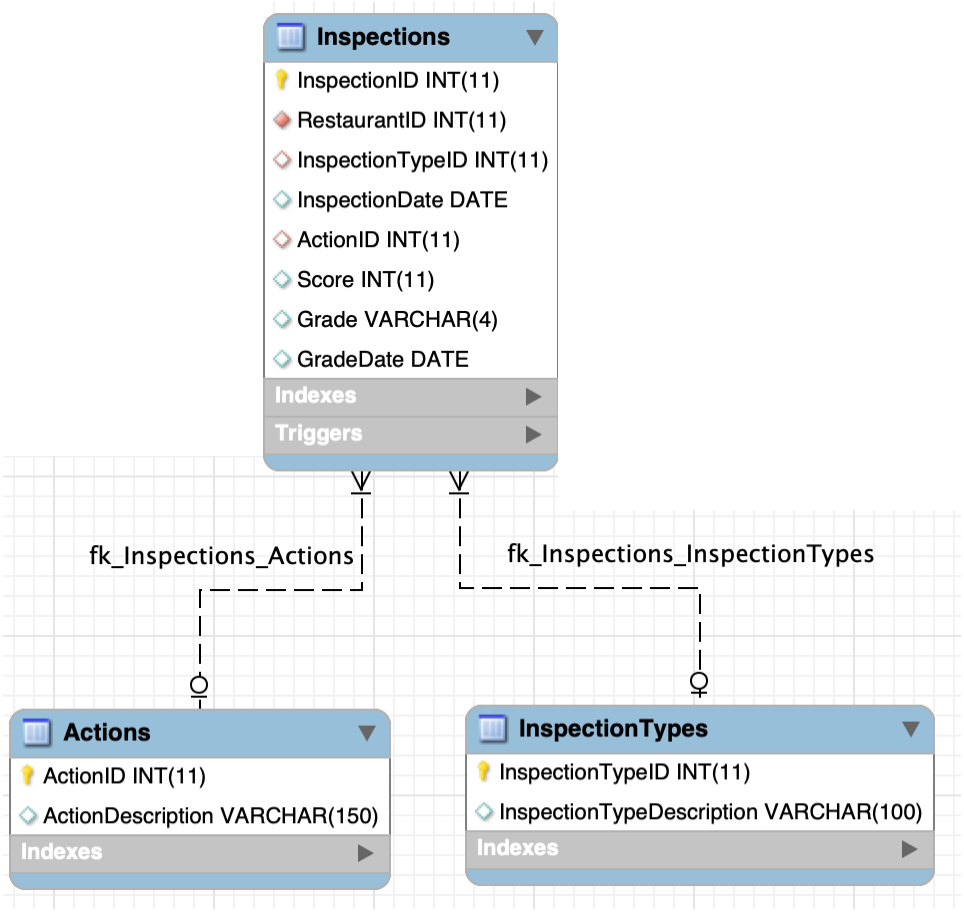


Make new tables to eliminate transitive dependencies

- Transitive dependency: If A→B and B→C, then A→C
- Identify by looking for dependencies on nonprime (nonprime means non-key) attributes
- Score → Grade

- Could make a new table to look up the Grade given the score
  - I choose not to here
  - Could also make a generated column for Score

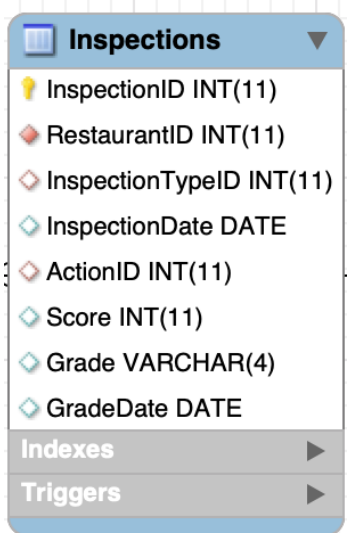
3NF: remove transitive from all tables



Another design choice:

- **I choose to create a look up table for Actions and for InspectionTypes (like we did for Cuisine)**

3NF: remove transitive from all tables



We still have a problem

InspectionID -> RestaurantID, InspectionDate, InspectionType

- But each row cannot be uniquely identified by just the PK
- The same inspection can cause multiple violation codes

The real key to uniquely identify rows is: InspectionID, ViolationCode
The surrogate key hides this problem!
Technically this table is in 2NF due to single attribute PK

This situation means RestaurantID, InspectionDate, and InspectionType are only dependent on part of the key (partially dependent) because they are not dependent on ViolationCode

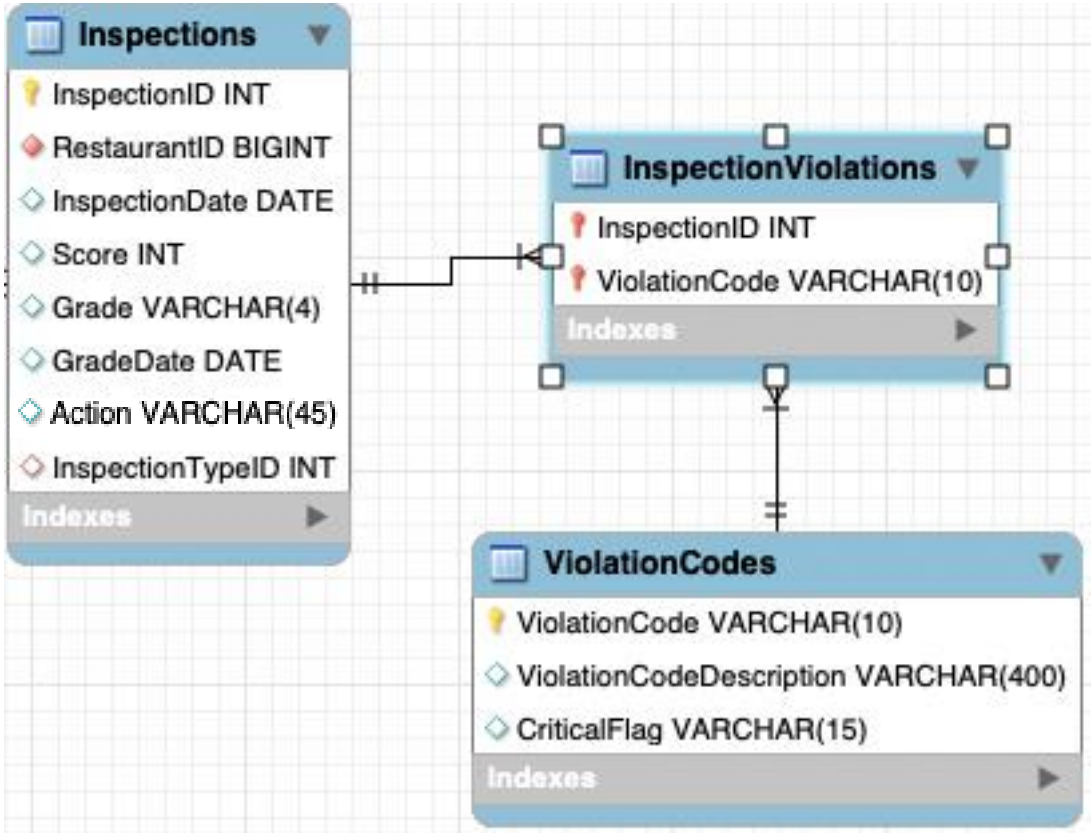
Solution: make a new table for InspectionViolations (this becomes a joining table from last class)

auto_increment makes unique InspectionID

InspectionID	RestaurantID	InspectionTypeID	InspectionDate	ViolationCode	Other columns
1	11111	5	2026-04-30	10A	…
2	11111	5	2026-04-30	6B	…
3	1111	5	2026-04-30	4G	…
4	2222	4	2025-12-31	9D	…
5	3333	1	2024-08-06	10A	…

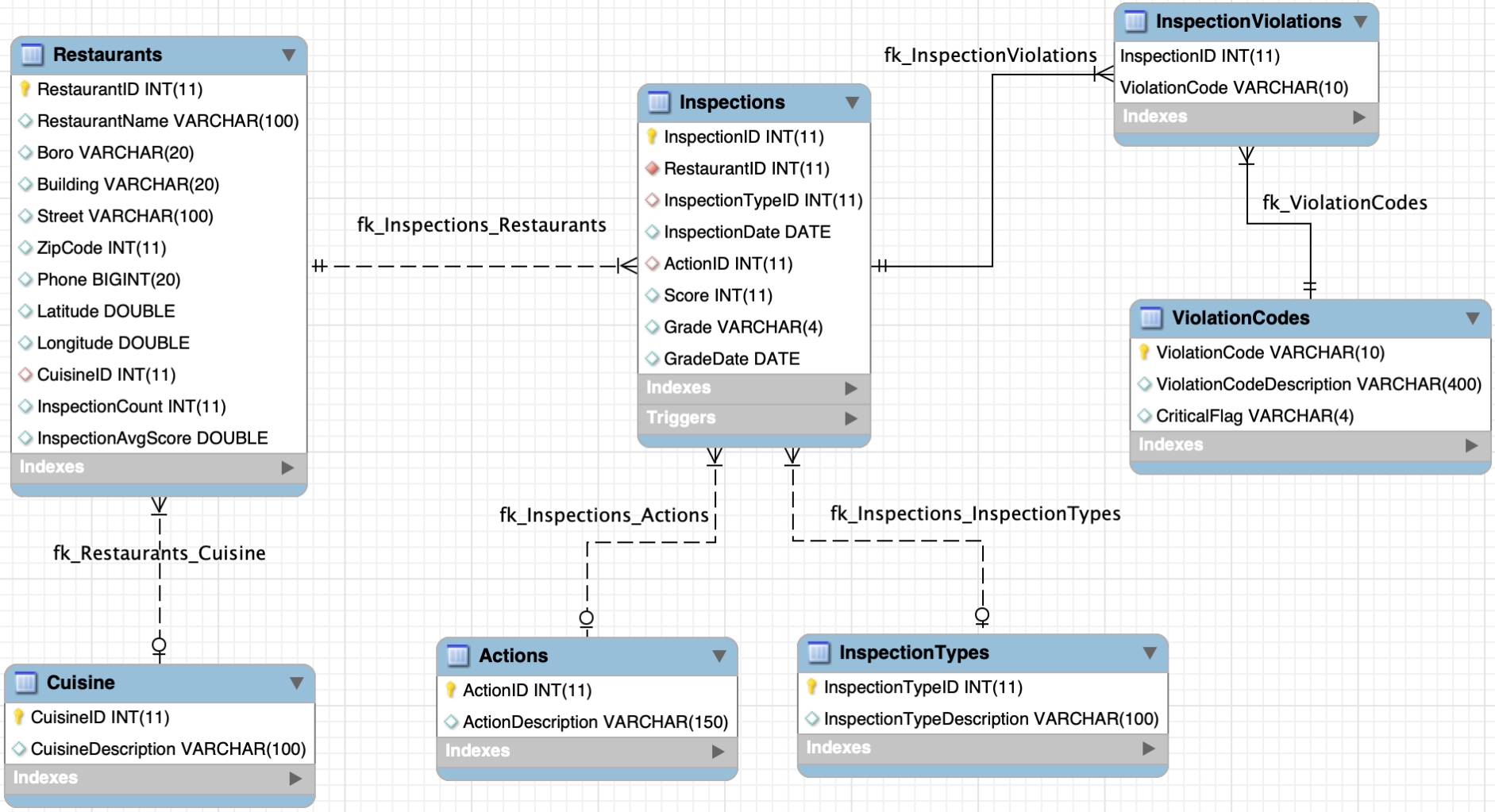
Same inspection

3NF: remove transitive from all tables



InspectionID	ViolationCode
1	10A
1	6B
1	4G
2	9D
3	10A

Final normalized design



Agenda

1. Theoretical normalization
2. Practical normalization

From last class

Database design goals:

- Well structured tables with minimal redundancy
- Store data facts one time

Pierson's simplified two-step view of database design:

Last class 1. Identify entities, attributes, and relations -> leads to database tables and keys

Today 2. Normalize tables -> check that your tables are well designed to prevent data anomalies (store data once!)

(There is also physical database design (e.g., how are database files organized), but we will discuss that soon)

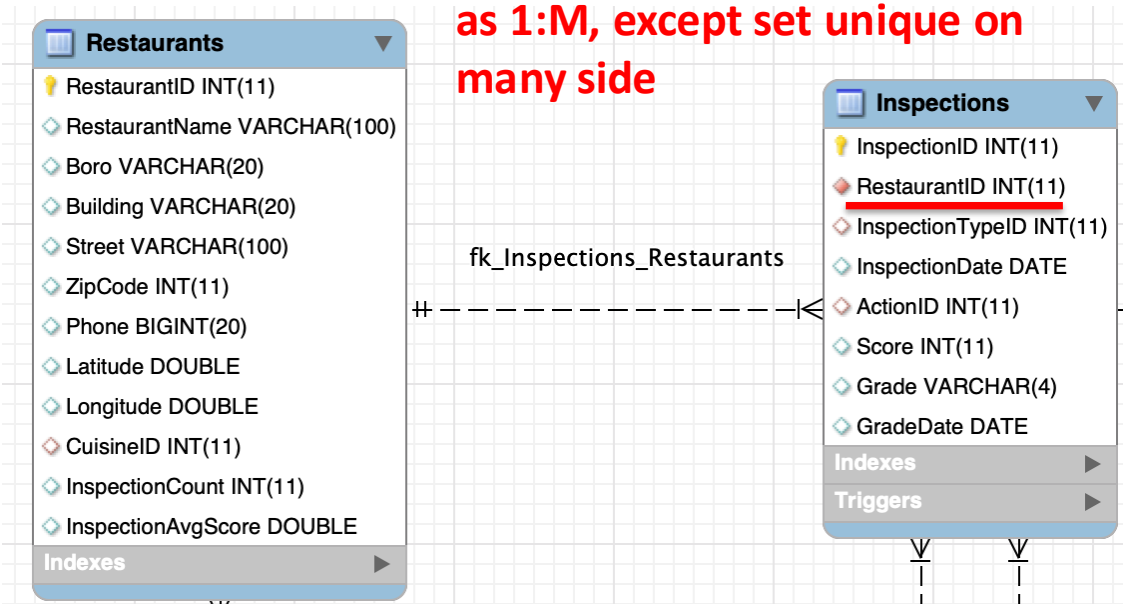
Done well, Step 1 makes Step 2 easy!

Step 1: Identify entities, attributes, and relations

Talk to people to understand the business rules

- Restaurants can be inspected multiple times
- Each inspection is only for one restaurant
- Entities: Restaurants and Inspections
- 1:M relationship between Restaurants and Inspections (use foreign key on the many side – Inspections here)

If 1:1 relationship, do the same as 1:M, except set unique on many side



Database theory says Inspections should have a compound key because each inspection can be identified by:

- **RestaurantID**
- **InspectionDate**
- **InspectionType**

In practice, we normally give each table a surrogate and then just move on to bigger issues

Also, it is extremely unlikely you will be given a large unnormalized table to begin in the real world! You'll need to talk to people to understand business rules

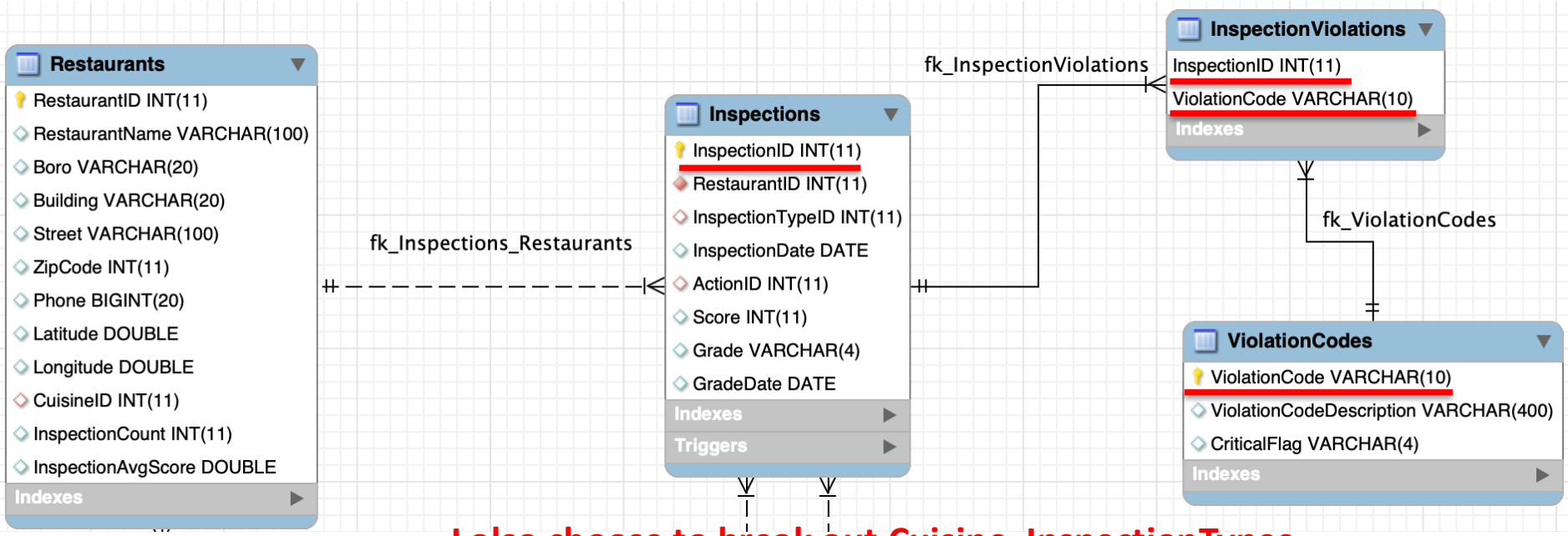
Step 1: Identify entities, attributes, and relations

Talk to people to understand the business rules

- Each Inspection can result in multiple violations
- Each Violation can appear in many Inspections
- Entities: Inspections and Violations
- M:N relations between Inspections and Violations (use joining table)

Create joining table

InspectionID	ViolationCode
1	10A
1	6B
1	4G
2	9D
3	10A



I also choose to break out Cuisine, InspectionTypes, and Actions into look up table

Step 2: normalize each table in turn

Normalize each table in turn

1NF: make sure in table form

- Almost assuredly already done
- Make sure every table has a key (probably surrogate in practice, compound in theory)

2NF: remove partial dependencies

- Not a problem if using surrogate keys on all (most) tables, already technically 2NF
- Use joining tables on M:N relationships to avoid sneaky dependencies

3NF: remove transitive dependencies

- Look for attributes that can be identified by a non-key (nonprime) attribute in the table
- Often will choose to leave a table slightly unnormalized for things like ZipCode identifying boro or State

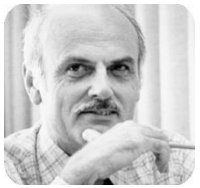
Summary

The key, the whole key, and nothing but the key, so help me Codd

1NF: "The key"
- Every table must have a key
- All values must be atomic

2NF: "The whole key"
- Every attribute depends on the WHOLE key
- No partial dependencies

3NF: "Nothing but the key"
- Every attribute depends on NOTHING BUT the key
- No transitive dependencies



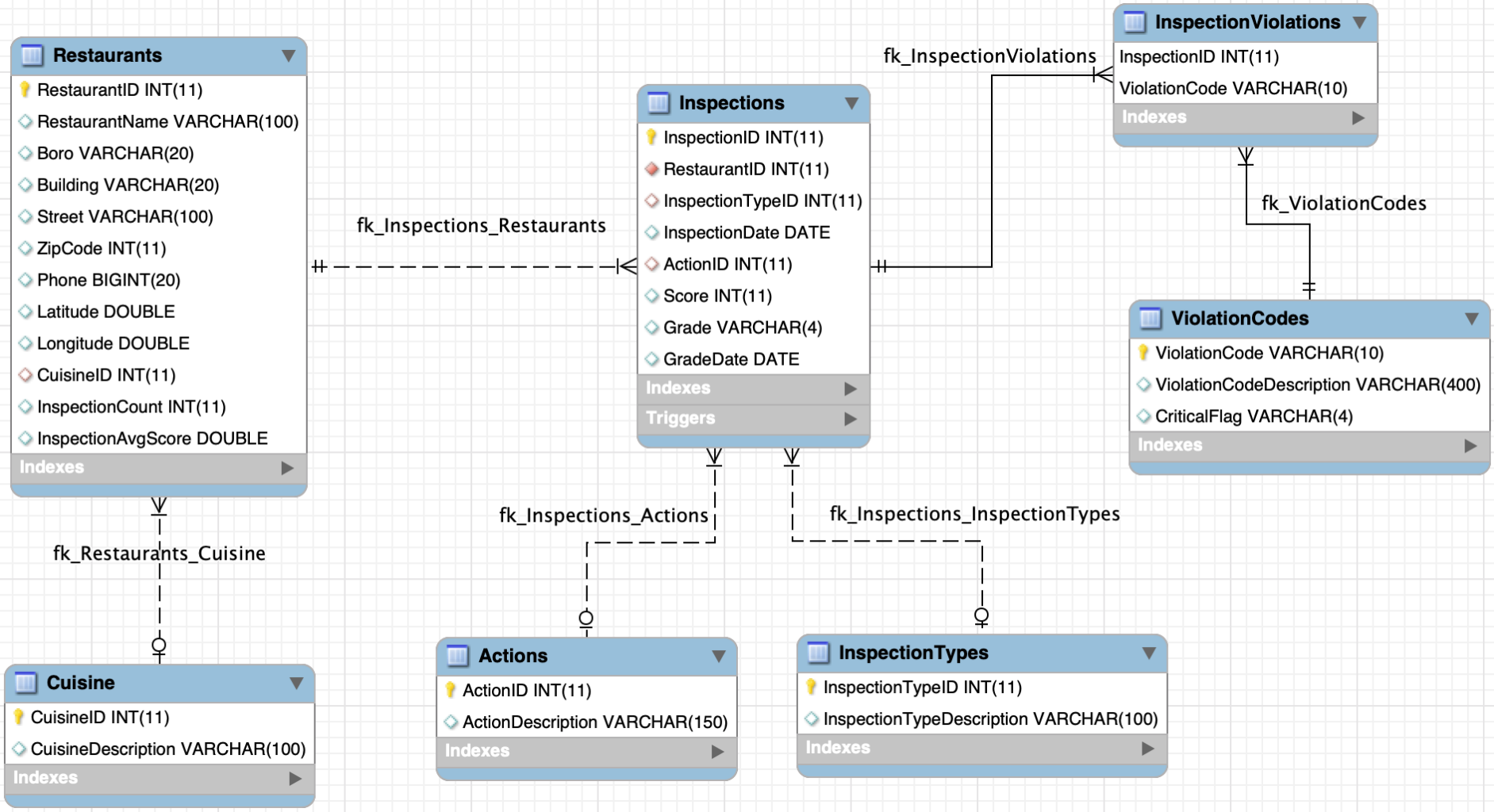
E.F. “Ted” Codd
Turing Award 1981

Normalization is valuable because it helps eliminate data redundancies

Other steps to consider after reaching 3NF

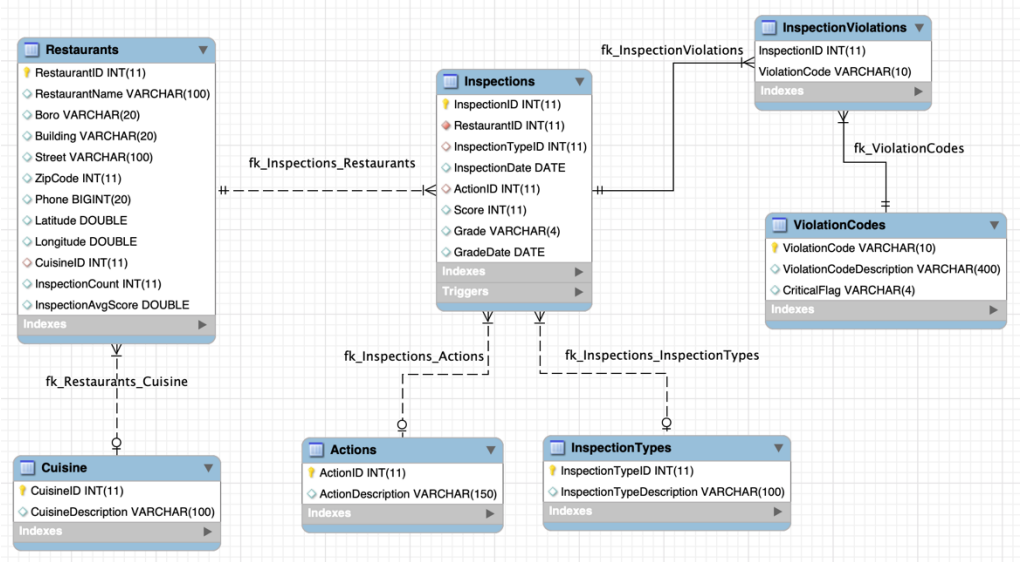
- Identify new attributes and new relationships (did we forget anything?)
- Refine attribute atomicity (will we ever need part of an attribute like first name if storing name as first and last name)
  - Atomic attribute: cannot be further subdivided
  - Atomicity: characteristic of an atomic attribute
- Evaluate using derived vs. stored attributes
- Consider foreign key requirements
  - Here we want Cuisine, Actions, and InspectionTypes to be selected from a set of known values
  - Create new tables for these, use PK as FK in Inspections

Final schema



How do we get inspection results?
Join tables!

JOIN tables to inspection results




```
6 •  USE nyc_data;
7 •  SELECT RestaurantName, InspectionDate, Grade, iv.ViolationCode, vc.ViolationCodeDescription
8        FROM Restaurants r JOIN Inspections i ON r.RestaurantID = i. RestaurantID
9            JOIN InspectionViolations iv on iv.InspectionID = i.InspectionID
10           JOIN ViolationCodes vc ON vc.ViolationCode = iv.ViolationCode
11       WHERE r.RestaurantName LIKE 'Morris Park Bake%'
12       ORDER BY i.InspectionDate DESC;
```

00% 41:2

Result Grid | Filter Rows: Search | Export:

RestaurantName	InspectionDa...	Grade	ViolationCode	ViolationCodeDescription
MORRIS PARK BAKE SHOP	2024-11-08	A	10F	Non-food contact surface improperly constructe…
MORRIS PARK BAKE SHOP	2024-11-08	A	08C	Pesticide not properly labeled or used by unlice…
MORRIS PARK BAKE SHOP	2024-11-08	A	06C	Food, supplies, and equipment not protected fro…
MORRIS PARK BAKE SHOP	2023-08-22	A	08C	Pesticide not properly labeled or used by unlice…
MORRIS PARK BAKE SHOP	2023-08-22	A	08A	Establishment is not free of harborage or conditi…
MORRIS PARK BAKE SHOP	2023-08-22	A	04L	Evidence of mice or live mice present in facility'…
MORRIS PARK BAKE SHOP	2023-02-03	P	02G	Cold food item held above 41° F (smoked fish a…
MORRIS PARK BAKE SHOP	2023-02-03	P	10F	Non-food contact surface improperly constructe…

Pros:

- **Data anomalies resolved**
- **Data stored one time**

Cons:

- **Queries are more complicated due to the need to join many tables**
- **Queries run slower due to joins**

Practice: identify entities and normalize tables

Soccer player database

PlayerID	Name	Team	TeamPhone	Position1	Position2	Position3
1	Pessi	Argentina	54-11-1000-1000	Striker	Forward	
2	Ricardo	Portugal	351-2-7777-7777	Right Midfield	Defending Midfielder	
3	Neumann	Brazil	55-21-4040-2020	Forward	Left Fullback	Right Fullback
4	Baily	Wales	44-29-1876-1876	Defending Midfielder	Striker	
5	Marioso	Argentina	54-11-1000-1000	Sweeper	Defending Midfielder	Striker
6	Pare	Brazil	55-21-4040-2020	Goalkeeper		

Based on Prof Charles Palmer lecture notes

Database anomalies

Soccer player database

We will soon call something Position1, Position2, Position3, … a repeating group

PlayerID	Name	Team	TeamPhone	Position1	Position2	Position3
1	Pessi	Argentina	54-11-1000-1000	Striker	Forward	
2	Ricardo	Portugal	351-2-7777-7777	Right Midfield	Defending Midfielder	
3	Neumann	Brazil	55-21-4040-2020	Forward	Left Fullback	Right Fullback
4	Baily	Wales	44-29-1876-1876	Defending Midfielder	Striker	
5	Marioso	Argentina	54-11-1000-1000	Sweeper	Defending Midfielder	Striker
6	Pare	Brazil	55-21-4040-2020	Goalkeeper		

Business rules

- **Each player uniquely identified by PlayerID (it is a Primary Key here)**
- **Each player plays for one team and can play one or more position**
- **Each team has one phone number**

Based on Prof Charles Palmer lecture notes

Insert anomaly: can not add data due to absence of other data

Soccer player database

PlayerID	Name	Team	TeamPhone	Position1	Position2	Position3
1	Pessi	Argentina	54-11-1000-1000	Striker	Forward	
2	Ricardo	Portugal	351-2-7777-7777	Right Midfield	Defending Midfielder	
3	Neumann	Brazil	55-21-4040-2020	Forward	Left Fullback	Right Fullback
4	Baily	Wales	44-29-1876-1876	Defending Midfielder	Striker	
5	Marioso	Argentina	54-11-1000-1000	Sweeper	Defending Midfielder	Striker
6	Pare	Brazil	55-21-4040-2020	Goalkeeper		
Ø	Ø	Iceland	54-12-5432-2345	Ø	Ø	Ø

Insert anomaly:

- **Can't add team (say Iceland) without adding a player for that team because PlayerID is Primary Key**
- **Also no consistency in position names**
  - **What if some teams call a Sweeper a Center Back**
  - **How would we know they are the same?**
- **What if a player can play more than three positions?**

Based on Prof Charles Palmer lecture notes

Update anomaly: must update multiple tuples for one change

Soccer player database

PlayerID	Name	Team	TeamPhone	Position1	Position2	Position3
1	Pessi	Argentina	54-11-1000-1000	Striker	Forward	
2	Ricardo	Portugal	351-2-7777-7777	Right Midfield	Defending Midfielder	
3	Neumann	Brazil	55-21-4040-2020	Forward	Left Fullback	Right Fullback
4	Baily	Wales	44-29-1876-1876	Defending Midfielder	Striker	
5	Marioso	Argentina	54-11-1000-1000	Sweeper	Defending Midfielder	Striker
6	Pare	Brazil	55-21-4040-2020	Goalkeeper		
Ø	Ø	Iceland	54-12-5432-2345	Ø	Ø	Ø

Update anomaly:

- **If team moves, must update TeamPhone for all players on that team**
- **Could lead to inconsistency if some team players are updated, but not all**

Based on Prof Charles Palmer lecture notes

Delete anomaly: unintended loss of data

Soccer player database

PlayerID	Name	Team	TeamPhone	Position1	Position2	Position3
1	Pessi	Argentina	54-11-1000-1000	Striker	Forward	
2	Ricardo	Portugal	351-2-7777-7777	Right Midfield	Defending Midfielder	
3	Neumann	Brazil	55-21-4040-2020	Forward	Left Fullback	Right Fullback
4	Baily	Wales	44-29-1876-1876	Defending Midfielder	Striker	
5	Marioso	Argentina	54-11-1000-1000	Sweeper	Defending Midfielder	Striker
6	Pare	Brazil	55-21-4040-2020	Goalkeeper		
Ø	Ø	Iceland	54-12-5432-2345	Ø	Ø	Ø

Delete anomaly:

- **If Ricardo retires, must remove from database**
- **If so, loose Portugal team data as well because Ricardo is the only Portugal player!**

How does this apply to our inspection database?

Based on Prof Charles Palmer lecture notes