CS 61:
Database Systems

Joins

Agenda

1. Joins
2. nyc_inspections schema
3. Joins on nyc_inspections
4. Conditional evaluation

With JOIN store data one time in multiple tables; combine to form larger table

instructor table

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000

teaches table

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-315	1	Spring	2018

Teaches table lists courses and sections that are taught by instructors

Book's schema also has additional table for courses (not shown)

With JOIN store data one time in multiple tables; combine to form larger table

instructor table

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000

teaches table

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-315	1	Spring	2018

SELECT *i.*, t.**
FROM *instructor i, teaches t* -- *cartesian product*
WHERE *i.ID = t.ID;* -- *filter cartesian product*

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017

If we kept data in one large table, there are several anomalies that can occur

Problems keeping one large table with many attributes

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018

Anomalies if keep one large table instead of multiple tables

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

Insert anomalies

Problems keeping one large table with many attributes

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018

Insert anomalies

- **If hire a new instructor, they do not show up in database until they teach a course**

Update anomalies

Problems keeping one large table with many attributes

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018

Update anomalies

- **If instructor gets a raise, must update salary in <u>all</u> rows for that instructor**
- **Can lead to inconsistent data!**
- **What is the instructor's true salary?**

Delete anomalies

Problems keeping one large table with many attributes

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018

Delete anomalies

- **If course is only taught one time and instructor taught only one course, if delete course, loose instructor too!**
- **If delete PHY-101, loose Einstein!**

We can avoid these anomalies by keeping data in multiple tables

instructor table

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000

teaches table

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-315	1	Spring	2018

Better to store data in multiple tables

Insert: new instructor can be added to database without teaching a class

Update: instructor gets a raise, only update one row in instructor table

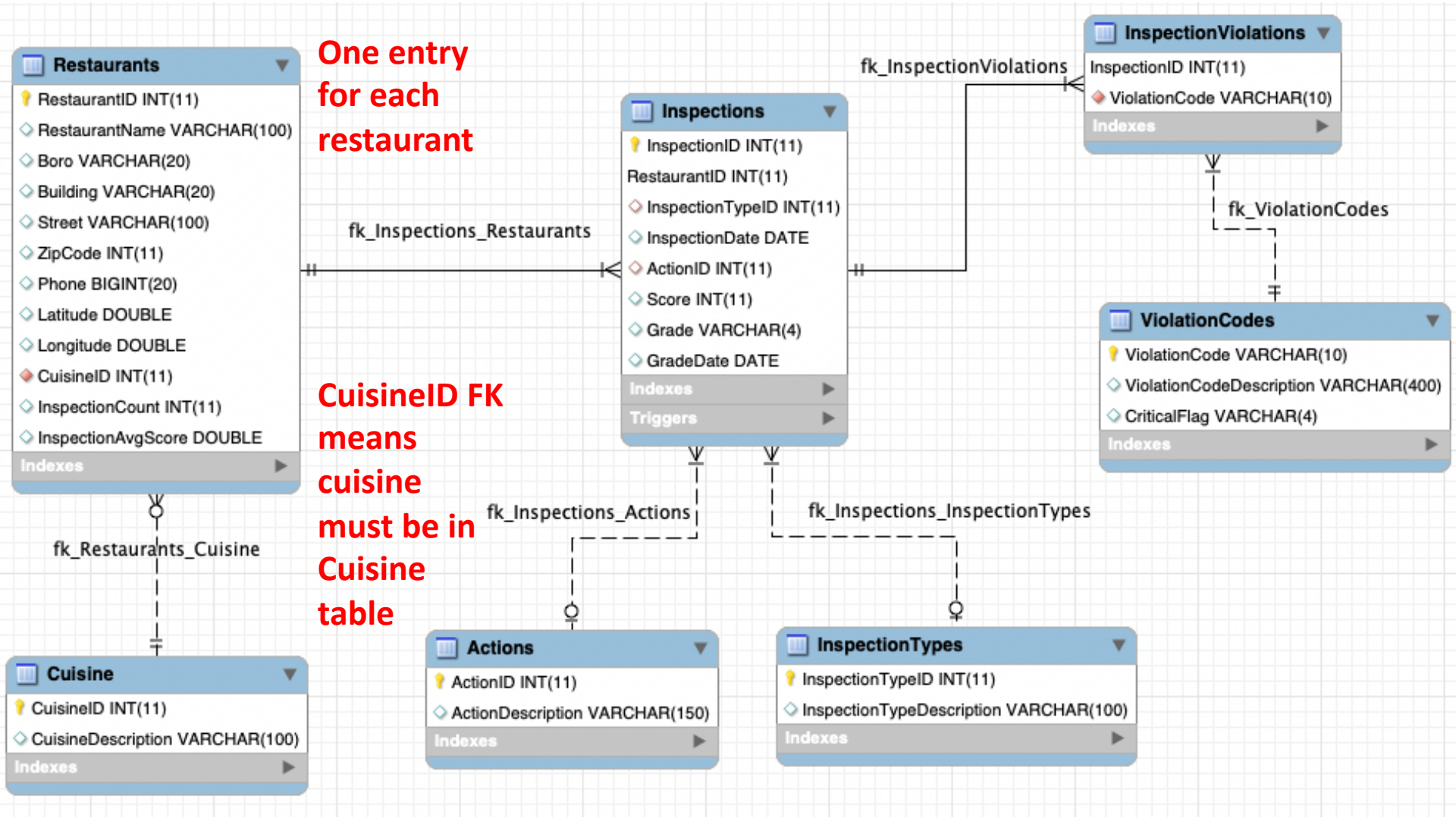
Delete: can delete course, instructor will still exist in instructor table

Agenda

1. Joins
2. nyc_inspections schema
3. Joins on nyc_inspections
4. Conditional evaluation

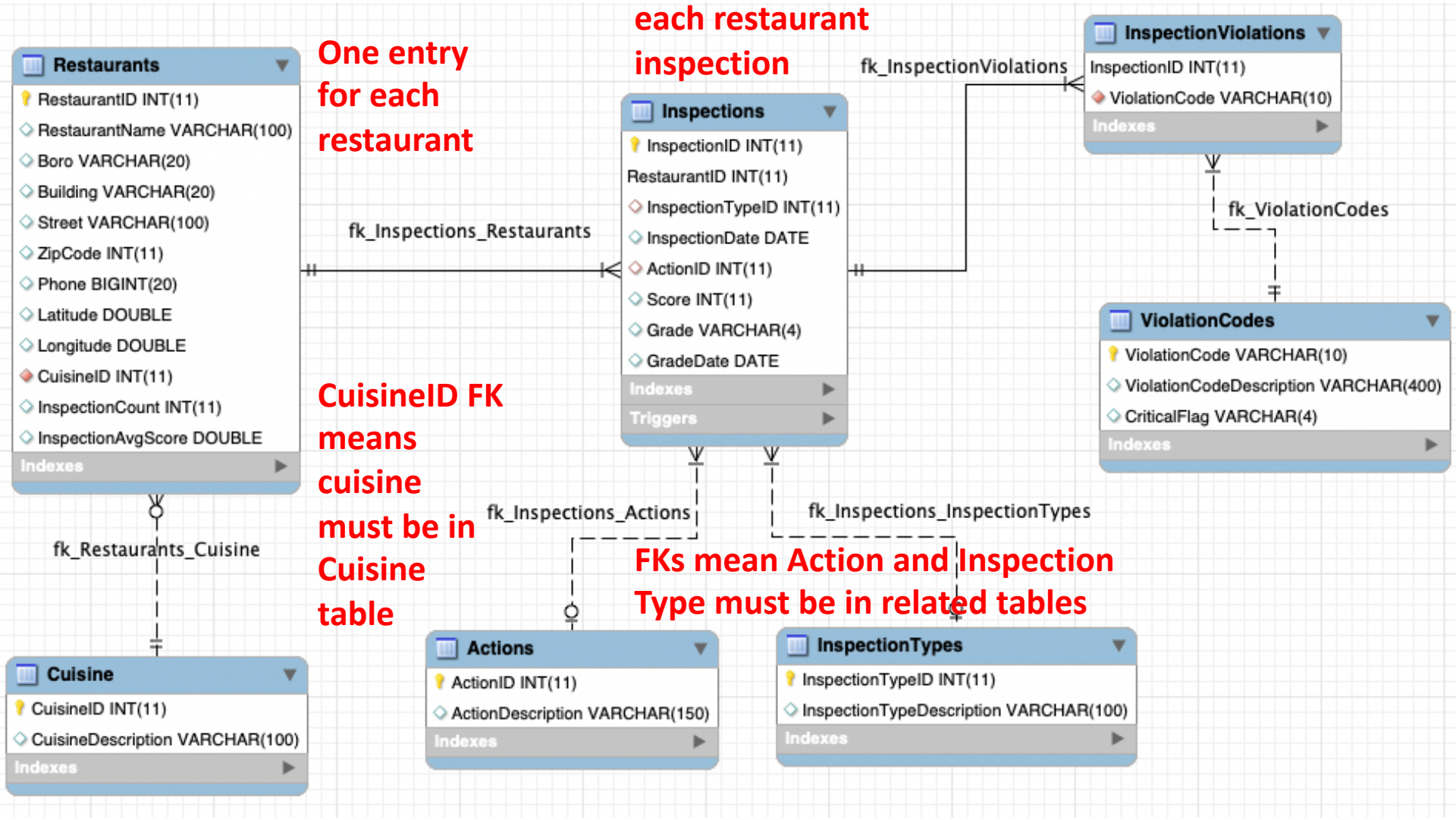
The old single table is now multiple related tables in nyc_inspections

use nyc_inspections;



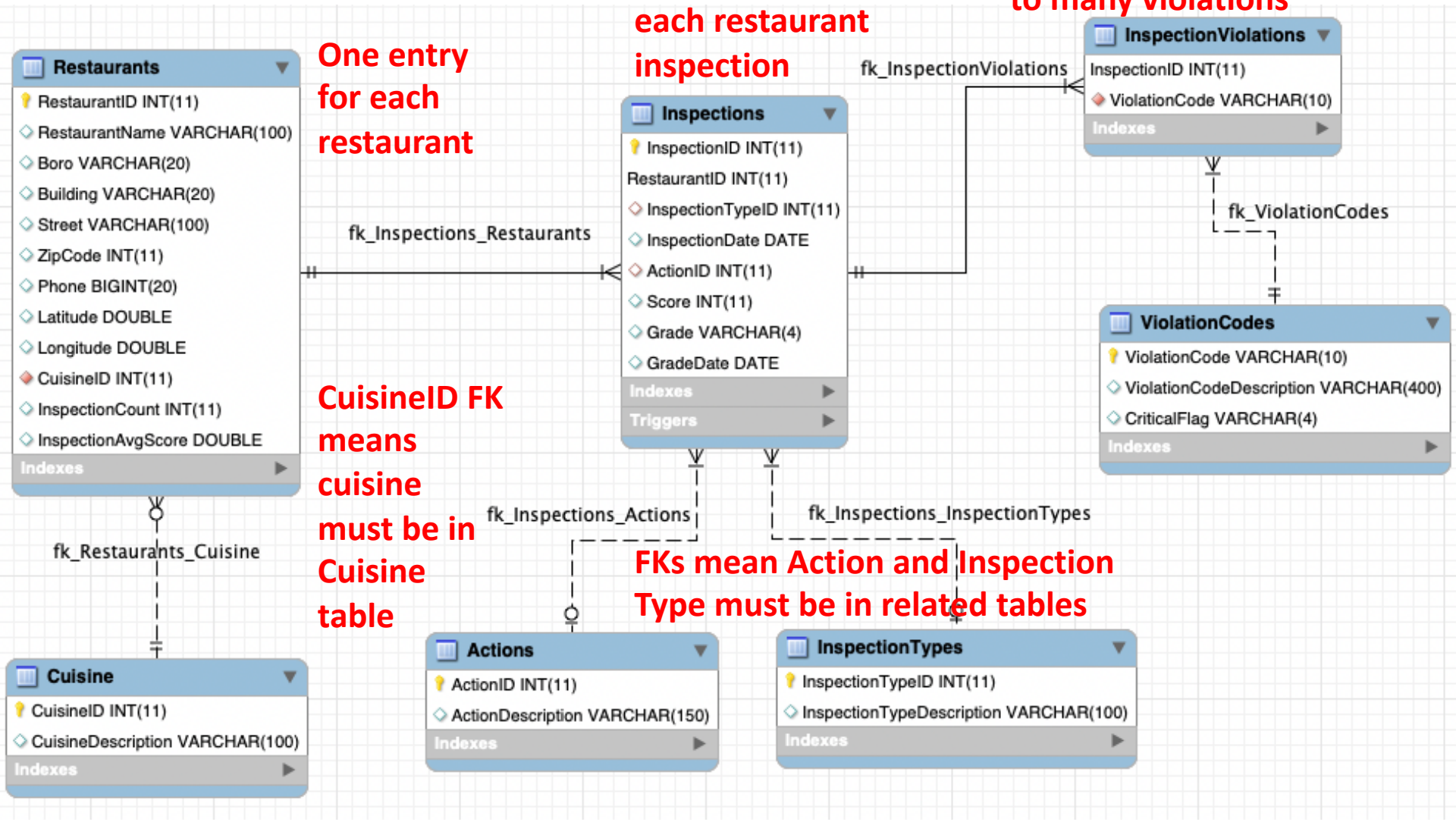
The old single table is now multiple related tables in nyc_inspections

use nyc_inspections;



The old single table is now multiple related tables in nyc_inspections

use nyc_inspections;



Agenda

1. Joins
2. nyc_inspections schema
3. Joins on nyc_inspections
4. Conditional evaluation

Recommended way to join is not in the WHERE clause, but with JOIN command

JOIN

Thus far we have joined relations by matching in the WHERE clause, but JOIN is preferred

Format:
SELECT $A_1, A_2, \ldots A_n$
FROM r_1 **{type} JOIN** r_2 **{type} JOIN .. {type} JOIN** r_n
where P ;
{type} = **[NATURAL | INNER | OUTER [LEFT RIGHT FULL]].** If type not specified, INNER

Joins store results in a temporary table in the database

Example: count how many time each bakery has been inspected

```
SELECT r.RestaurantID, RestaurantName, count(*)
FROM Restaurants r, Inspections i
WHERE r.RestaurantID = i.RestaurantID
AND r.CuisineID = 5 - - look up bakery ID in Cuisine table
GROUP BY RestaurantID;
```

Our previous way (old style join):
Join is done in the WHERE clause
Must specify which table attribute from

Both do the same thing!

Preferred way:

Recommended way:
Join is done in the FROM clause using JOIN; implicitly an INNER join

```
SELECT r.RestaurantID, RestaurantName, count(*)
FROM Restaurants r JOIN Inspections i ON r.RestaurantID = i.RestaurantID
WHERE r.CuisineID = 5
GROUP BY RestaurantID;
```

Can still use WHERE to limit results

Recommended way to join is not in the WHERE clause, but with JOIN command

JOIN

Thus far we have joined relations by matching in the WHERE clause, but JOIN is preferred

Format:
SELECT $A_1, A_2, \ldots A_n$
FROM r_1 **{type} JOIN** r_2 **{type} JOIN .. {type} JOIN** r_n
where P ;
{type} = **[NATURAL | INNER | OUTER [LEFT RIGHT FULL]].** If type not specified, INNER

Example: count how many time each bakery has been inspected

```
SELECT r.RestaurantID, RestaurantName, count(*)
FROM Restaurants r, Inspections i
WHERE r.RestaurantID = i.RestaurantID
AND r.CuisineID = 5 - - look up bakery ID in Cuisine table
GROUP BY RestaurantID;
```

Preferred way:

```
SELECT RestaurantID, RestaurantName, count(*)
FROM Restaurants r NATURAL JOIN Inspections i
WHERE CuisineID = 5
GROUP BY RestaurantID;
```

Could also do a NATURAL JOIN

- **No need to tell which attributes to match for join (uses attributes with same name to join)**
- **No duplicate attributes in result**

I prefer using JOIN ON (last slide)
I know for sure what attributes are used for join (or at least use JOIN USING)

INNER join only returns rows if comparison attribute is in both tables

INNER JOIN

TableA

ID	A_1
1	m
2	**n**
4	o

TableB

ID	A_2
2	**p**
3	q
5	r

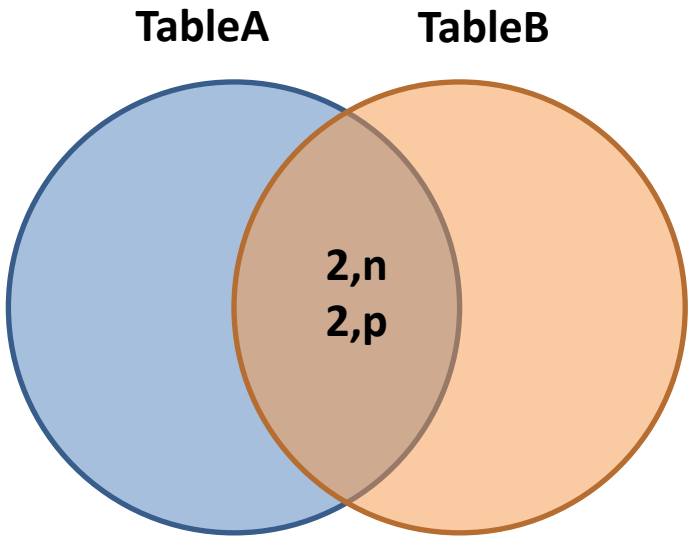
Rows returned with attributes from both tables if match between values in comparison columns

```
SELECT *
FROM TableA a JOIN TableB b
ON a.ID=b.ID
```

ID of 2 is in both tables so it is returned, others are not

Result has attributes from both tables (A_1 and A_2) and duplicate ID

Rows 1 and 4 from TableA and rows 3 and 5 from TableB not returned



Result (temp table)

ID	A_1	ID	A_2
2	n	2	p

NATURAL JOIN omits duplicate attributes (could also pick in SELECT)

ID	A_1	A_2
2	n	p

Adapted from: https://www.w3resource.com/slides/sql-joins-slide-presentation.php

LEFT OUTER JOIN returns all rows from the left table

LEFT [OUTER] JOIN

TableA

ID	A_1
1	m
2	n
4	o

TableB

ID	A_2
2	p
3	q
5	r

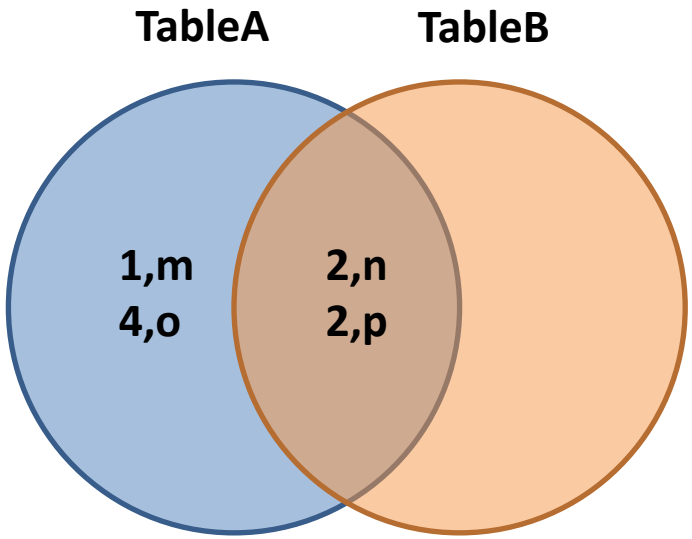
All rows from TableA returned

```
SELECT *
FROM TableA a LEFT JOIN TableB b
ON a.ID=b.ID
```

ID of 2 is in both tables so it is returned

All rows from left table (TableA) as written in command are returned

3 and 5 in TableB not returned because those keys not in TableA



Result (temp table)

ID	A_1	ID	A_2
2	n	2	p
1	m	NULL	NULL
4	n	NULL	NULL

Adapted from: https://www.w3resource.com/slides/sql-joins-slide-presentation.php

RIGHT OUTER JOIN returns all rows from the right table

RIGHT [OUTER] JOIN

TableA

ID	A_1
1	m
2	**n**
4	o

TableB

ID	A_2
2	**p**
3	**q**
5	**r**

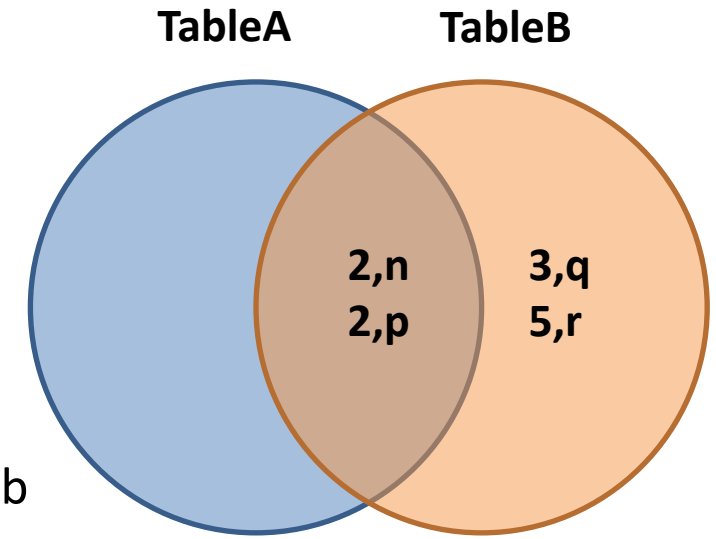
All rows from TableB returned

```
SELECT *
FROM TableA a RIGHT JOIN TableB b
ON a.ID=b.ID
```

ID of 2 is in both tables so it is returned

All rows from right table (TableB) as written in command are returned

1 and 4 in TableA not returned because those keys not in TableB



Result (temp table)

ID	A_1	ID	A_2
2	n	2	p
NULL	NULL	3	q
NULL	NULL	5	5

Adapted from: https://www.w3resource.com/slides/sql-joins-slide-presentation.php

FULL OUTER JOIN returns all rows from both tables

FULL [OUTER] JOIN

TableA

ID	A_1
1	m
2	n
4	o

TableB

ID	A_2
2	p
3	q
5	r

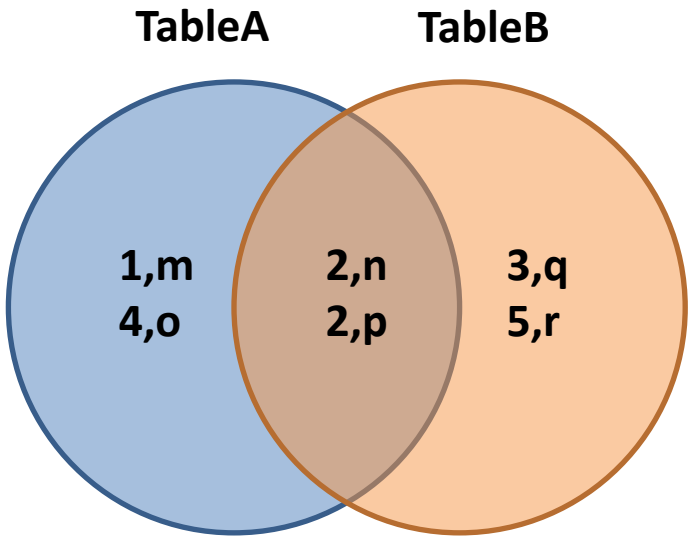
All rows from both tables returned

```
SELECT *
FROM TableA a FULL JOIN TableB b
ON a.ID=b.ID
```

ID of 2 is in both tables so it is returned

All rows from both tables are returned

NOTE: MySQL does not support FULL OUTER JOIN



Result (temp table)

ID	A_1	ID	A_2
2	n	2	p
1	m	NULL	NULL
4	n	NULL	NULL
NULL	NULL	3	q
NULL	NULL	5	5

Adapted from: https://www.w3resource.com/slides/sql-joins-slide-presentation.php

Practice

Use nyc_inspections

You've opened a new fruit/vegetable restaurant in Manhattan called 'Tim's Tasty Treats' (keep the apostrophe in the name!):

- Insert a new row in your Restaurants table for this restaurant
  - Set the RestaurantID to 1111
  - Set the CuisineID to the proper value for a fruits/vegetables restaurant
  - You can set address, phone, lat/long to NULL (or another value)
- See how many other fruit/vegetable restaurants there are (your competition)
- Count how many times each fruit/vegetable restaurant has been inspected, include:
  - RestaurantID
  - RestaurantName
  - Count of inspections
  - Make sure Tim's restaurant is on the list and shows zero inspections! (note: this is tricky!)

Agenda

1. Joins
2. nyc_inspections schema
3. Joins on nyc_inspections
4. Conditional evaluation

IF allows conditional evaluation, giving one of two values

IF command

If expr results in true or not null, return true value else return false value

Format:

SELECT IF (expr, true value, false value) **AS** name

Like a lambda expression in many programming languages

Practice: Some restaurant inspection grades are NULL
If Grade is null, return 'N/A' else return Grade

Note: if attribute could come from more than relation, identify which one

```
use nyc_inspections;
SELECT i.RestaurantID, RestaurantName, InspectionDate, Score,
      IF(Grade IS NULL,'N/A',Grade) AS Grade, GradeDate
FROM Inspections i JOIN Restaurants r on i.RestaurantID = r.RestaurantID
WHERE r.RestaurantID = 30075445
ORDER BY InspectionDate;
```

See day5.sql

COALESCE returns the first non-null value in a list of arguments

COALESCE

Format:

SELECT COALESCE(value$_1$, value$_2$, … value$_n$) **AS** name

value$_x$ can be an attribute or literal
If value$_1$ is NULL, SQL tries value$_2$, …
Returns first non-null value
If all values are NULL, returns NULL

Given a table of customer orders

- **Try using State as Location**
- **If State is NULL, try Country**
- **If Country is NULL, use 'N/A'**

Example:

SELECT OrderID, **COALESCE**(State, Country, 'N/A') **AS** Location
FROM Orders

CASE provides if-then-else logic in one of two formats

CASE

Value of attribute

```
SELECT CASE attribute
   WHEN value1 THEN result1
   WHEN value2 THEN result2
   [ELSE else result]
END AS AttributeName
```

OR

No attribute here, it is in expression

```
SELECT CASE
   WHEN expr1 THEN result1
   WHEN expr2 THEN result2
   [ELSE else result]
END AS AttributeName
```

Don't forget END!

Practice: Categorize restaurants as having infrequent (<10), moderate (10-15), or frequent (>10) inspections

See day5.sql

```
SELECT i.RestaurantID, RestaurantName, count(*) AS `Total Inspections`,
      CASE
            WHEN count(*) < 10 THEN 'Infrequent inspections '
            WHEN count(*) BETWEEN 10 AND 15 THEN 'Moderate inspections'
            ELSE 'Frequent inspections'
      END AS Frequency
FROM Inspections i, Restaurants r
WHERE i.RestaurantID = r.RestaurantID
GROUP BY i.RestaurantID;
```

Can use BETWEEN x and y; or

Could have used count(*) >10 AND count(*) > 15

Practice

CASE practice

We can combine CASE with aggregate operations. For each restaurant provide the number of times an inspection resulted in each possible letter Grade

- First determine the grades possible (or at least those than have been given)
- Then on one line provide the number of times each restaurant received a distinct grade (e.g. RestaurantID, RestaurantName, A, B, C…, NULL, Total)
- Output should look like this:

RestaurantID	RestaurantName	A	B	C	G	N	P	Z	NULL	Total
30075445	MORRIS PARK BAKE SHOP	4	0	0	0	0	0	0	2	6
30112340	WENDY'S	5	0	0	0	0	0	0	4	9
30191841	DJ REYNOLDS PUB AND RESTAU…	3	0	0	0	0	0	0	1	4
40356018	RIVIERA CATERERS	3	0	0	0	0	0	0	1	4
40356151	BRUNOS ON THE BOULEVARD	2	0	0	0	0	0	0	2	4
40356483	WILKEN'S FINE FOOD	3	0	0	0	0	0	0	0	3
40356731	TASTE THE TROPICS ICE CREAM	3	0	0	0	0	0	0	0	3
40357217	WILD ASIA	3	0	0	0	0	0	0	0	3
40359480	1 EAST 66TH STREET KITCHEN	3	0	0	0	0	0	0	0	3