I DON'T ALWAYS LIST THE COLUMNS
IN MY SELECT STATEMENT
BUT WHEN I DO, ITS
WHEN ANOTHER
DBA IS WATCHING

Module 2-3

Joins

Objectives

- Keys (Primary, Natural, Surrogate, Foreign)
- Cardinality (1-1, 1-M, M-M)
- SQL Joins
- Unions

Amazon Scenario

Users table

Records for each user

Shipping_Addresses table

Shipping Addresses for each user

Purchases table

User purchases information

Products table

Product Data

Keys

In a relational database, all rows must be unique. The column or combination of columns that make it unique are referred to as **key(s)**.

- **Natural Key**: From real world data, SSN's, customer account numbers, driver license numbers
- **Surrogate Key**: Keys artificially created by an application to make a row unique

Keys

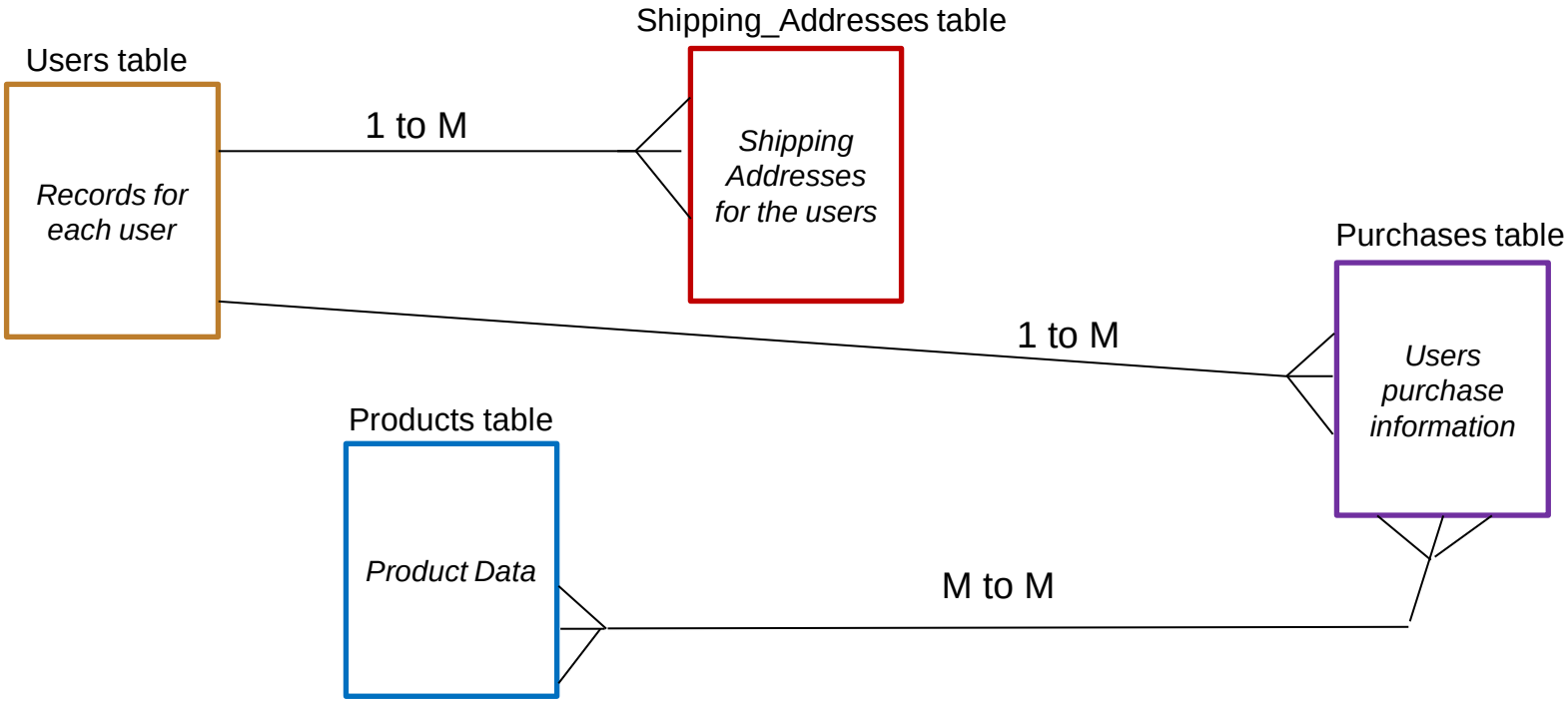
In a relational database, all rows must be unique. The column or combination of columns that make it unique are referred to as **key(s)**.

- **Primary Key**: column or columns in a table that uniquely identify the row. These cannot be duplicated.
  - If you say that SSN is your key, there cannot be more than one row with the same SSN.
- **Foreign Key**: A key that exists in another table, in which the latter is the primary key.

Cardinality

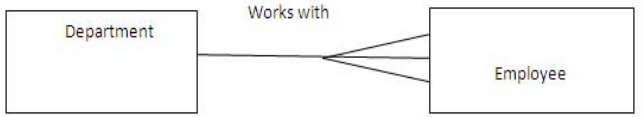
- Describes relationship between two tables
- Relationship between a row in one table and a row of another table.
- Options are one or many
- 1 to 1, 1 to M, M to M

Amazon Scenario

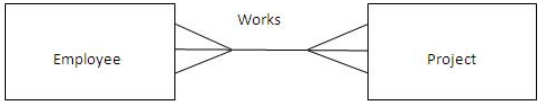


Other examples

a one-to-one relationship



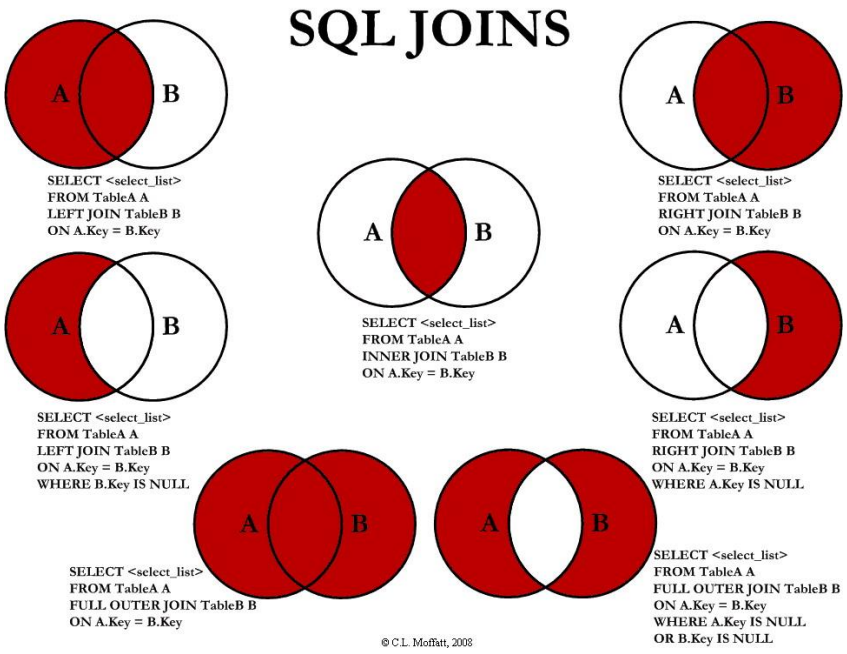
a one-to-many relationship



a many-to-many relationship

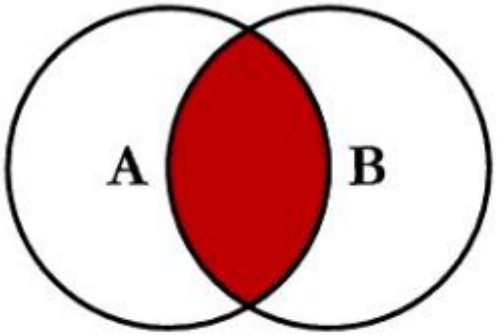
Joins

Joins in SQL allow us to pull in data from several tables.



Joins : Inner Join

An inner join returns the rows in Table A that has a matching key value in Table B, the Venn Diagram representation is as follows:




```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

Joins : Inner Join Example

Consider the following example: **We need a SQL query that returns all the languages spoken in South Africa. In my result, I want to see the full country name (not the code) followed by a language spoken in the country**..

*	countrycode	language	isofficial	percentage
1	AFG	Pashto	true	52.4
2	NLD	Dutch	true	95.6
3	ANT	Papiamento	true	86.2
4	ALB	Albaniana	true	97.9
5	DZA	Arabic	true	86.0
6	ASM	Samoan	true	90.6
7	AND	Spanish	false	44.6
8	AGO	Ovimbundu	false	37.2

- The countrylanguage table contains the list of languages by country code…
  but it is missing the full country name.
- The country table contains the list of all countries, but it has no data for languages.

*	code	name	continent	region	surfacearea	indepyear	population	li
1	AFG	Afghanistan	Asia	Southern and Central Asia	652090.0	1919	22720000	
2	NLD	Netherlands	Europe	Western Europe	41526.0	1581	15864000	
3	ANT	Netherlands Antilles	North America	Caribbean	800.0	(null)	217000	
4	ALB	Albania	Europe	Southern Europe	28748.0	1912	3401200	
5	DZA	Algeria	Africa	Northern Africa	2381741.0	1962	31471000	

- What we need to do is combine both tables:
  - Fortunately, these two tables are “related” via the country code value. On country, this is called code and on the countrylanguage table this is referred to as countrycode.

Joins : Inner Join Example

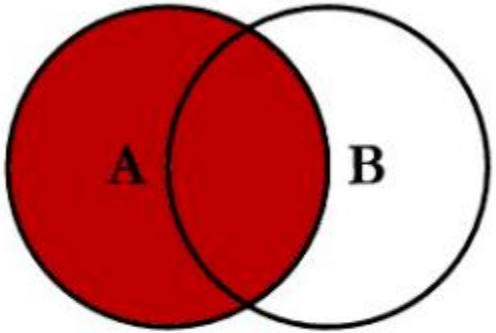
We can combine all of these facts to write a query that combines these two tables:

```
SELECT language
FROM countrylanguage A
INNER JOIN country B
ON A.countrycode = B.code
WHERE B.name = 'South Africa';
```

	language
1	Zulu
2	Xhosa
3	Afrikaans
4	Northsotho
5	English
6	Tswana
7	Southsotho
8	Tsonga
9	Swazi
10	Venda
11	Ndebele

Joins : Left Outer Join

The Left Outer Join returns all the rows on the “left” side table of the join, it will attempt to match to the right side. If there is match… If it can’t find a match it includes it in the result, but with NULL values.




```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

Joins : Left Outer Join Example

For this example, **I have manually removed all the rows on the countrylanguage table for China and Switzerland**.

```
SELECT A.name, B.language
FROM COUNTRY A
LEFT OUTER JOIN countrylanguage B
ON A.code = B.countrycode
WHERE name IN ('Switzerland', 'China', 'Belize');
```

*	name	language
1	Belize	English
2	Belize	Maya Languages
3	Belize	Garifuna
4	Switzerland	(null)
5	China	(null)

Note that the country codes for China and Switzerland don't exist, so the Left Outer Join instead creates these NULL placeholders.

Joins : Left Outer Join vs Inner Join

With the same data set as the previous slide, let's compare the LEFT OUTER vs an INNER.

```
SELECT A.name, B.language
FROM COUNTRY A
INNER JOIN countrylanguage B
ON A.code = B.countrycode
WHERE name IN ('Switzerland', 'China', 'Belize');
```

*	name	language
1	Belize	English
2	Belize	Maya Languages
3	Belize	Garifuna

With the INNER JOIN, the rows for which there are no matches on the key are dropped from the final result set.

```
SELECT A.name, B.language
FROM COUNTRY A
LEFT OUTER JOIN countrylanguage B
ON A.code = B.countrycode
WHERE name IN ('Switzerland', 'China', 'Belize');
```

*	name	language
1	Belize	English
2	Belize	Maya Languages
3	Belize	Garifuna
4	Switzerland	(null)
5	China	(null)

Unions

A union is a combination of two result sets. The following pattern is used:

[SQL Query 1]

UNION

[SQL Query 2]

Unions Example

Consider the following query:

```
SELECT countrycode, language, percentage FROM countrylanguage where language = 'Danish'
UNION
SELECT countrycode, language, percentage FROM countrylanguage where language = 'Swedish'
ORDER BY countrycode;
```

*	countrycode	language	percentage
1	DNK	Danish	93.5
2	FRO	Danish	0.0
3	NOR	Danish	0.4
4	GRL	Danish	12.5
5	DNK	Swedish	0.3
6	SWE	Swedish	89.5
7	NOR	Swedish	0.3
8	FIN	Swedish	5.7

Result of query first query (language = ‘Danish’)

Result of query second query (language = ‘Swedish’)

Union All

Suppose we changed the previous to only return the language instead:

```
SELECT language FROM countrylanguage where language = 'Danish'
UNION
SELECT language FROM countrylanguage where language = 'Swedish'
ORDER BY language
```

Note that the query only returns the unique values associated with countrylanguage

Union All

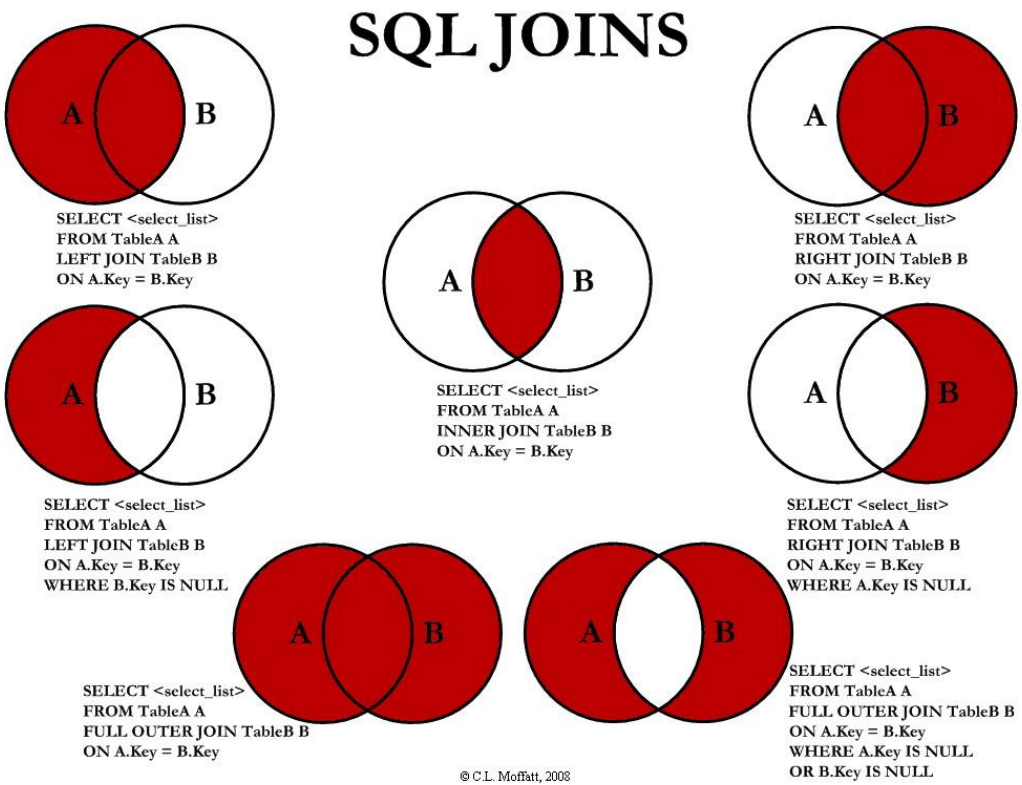
In situations like this, we can override this behavior by specifying UNION ALL instead:

```
SELECT language FROM countrylanguage where language = 'Danish'
UNION ALL
SELECT language FROM countrylanguage where language = 'Swedish'
ORDER BY language
```

*	language
1	Danish
2	Danish
3	Danish
4	Danish
5	Swedish
6	Swedish
7	Swedish
8	Swedish

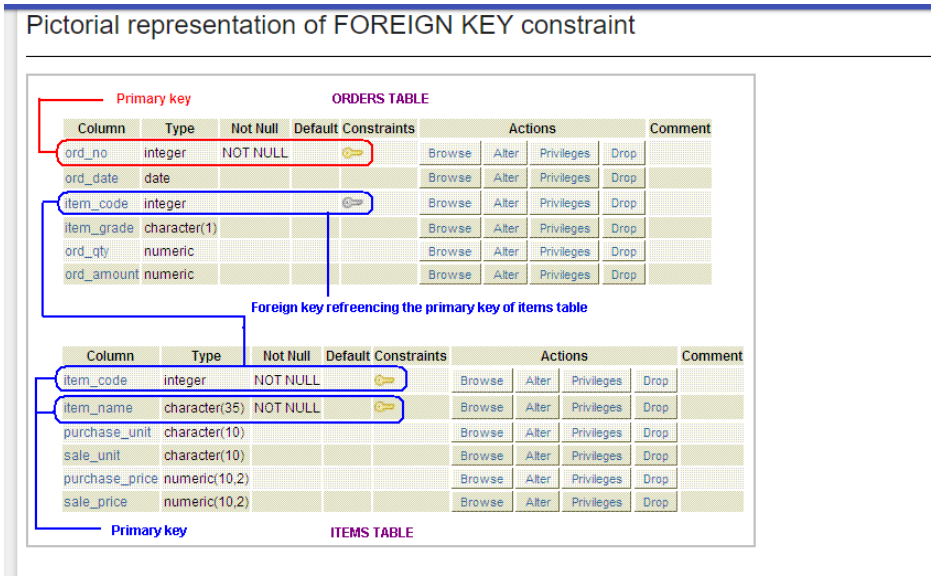
Duplicates are now not taken into account.

Joins


SQL JOINS
A
B
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
© C.L. Moffatt, 2008

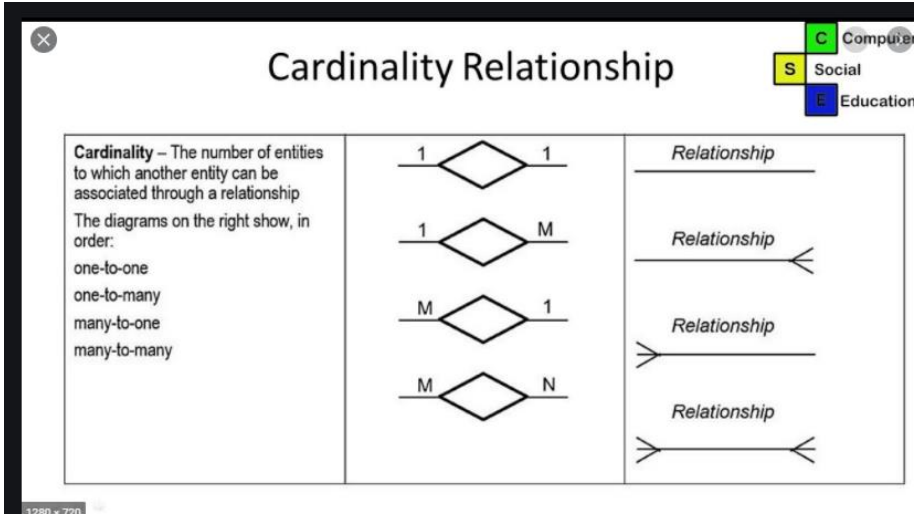
Objectives

- Keys (Primary, Natural, Surrogate, Foreign)



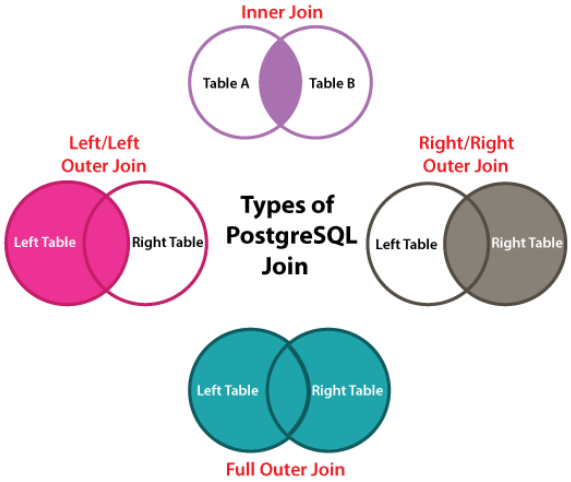
Objectives

- Keys (Primary, Natural, Surrogate, Foreign)
- Cardinality (1-1, 1-M, M-M)



Objectives

- Keys (Primary, Natural, Surrogate, Foreign)
- Cardinality (1-1, 1-M, M-M)
- SQL Joins

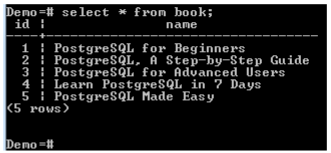


Objectives

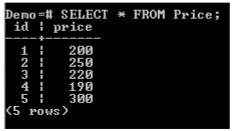
- Keys (Primary, Natural, Surrogate
- Cardinality (1-1, 1-M, M-M)
- SQL Joins
- Unions

We have a database named Demo with the following tables:

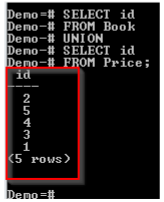
Book:



Price:



Let us run the following command:

```
SELECT id
FROM Book
UNION
SELECT id
FROM Price;
```

The command will return the following: