Practical Aspects of Database Design

L5 - Database Session II

Stevens Institute of Technology

Primary Key

What is it

- A primary key is a column or a group of columns that is used to identify a row uniquely in a table
- a primary key constraint is the combination of a not-null constraint and a UNIQUE constraint
- A table can have one and only one primary key

How to use it

- Define primary key when creating the table

```
CREATE TABLE table_name (
column_1 data_type PRIMARY KEY,
column_2 data_type );
```

Primary Key

How to use

- Define primary key when changing the existing table structure

```
ALTER TABLE table_name
ADD PRIMARY KEY primary_key;
```

- add auto-incremented primary key to existing table

```
ALTER TABLE table_name
ADD COLUMN col_name SERIAL PRIMARY KEY
primary_key;
```

- Remove primary key

```
ALTER TABLE table_name
DROP CONSTRAINT  constraint_pkey;
```

Foreign Key

What is it

- A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY(or at least a candidate key) in another table
- It is used to prevent actions that would destroy links between tables. It also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to

How does it work

- The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.
- A table can have multiple foreign keys depending on its relationships with other tables

Foreign Key

An example

- The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.
- The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

"Persons" table:

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

"Orders" table:

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Foreign Key

- Define simple PostgreSQL foreign key constraint

```
CREATE TABLE col_1 data_type, ...
FOREIGN KEY (col_name)
REFERENCES parent_table(primary_key) ;
```

- Define a group of columns as a foreign key

```
CREATE TABLE col_1 data_type, ...
FOREIGN KEY (col_name1, col1_name2)
REFERENCES parent_table (primary_key1, primary_key2) ;
```

Foreign Key

- Add foreign key constraint to existing table

```
ALTER TABLE table_name
ADD FOREIGN KEY (col_name)
REFREENCES parent_table (primary_key);
```

- To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name
FOREIGN KEY (col_name)
REFREENCES parent_table (primary_key);
```

Foreign Key

- Drop existing foreign key constraint

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_fkey;
```

Joining Multiple Tables Overview

- Natural join: joins two or more tables using implicit join condition based on the common column names in the joined tables.
- Full Outer Join: uses the full join to find a row in a table that does not have a matching row in another table.
- Inner Join: selects rows from one table that have the corresponding rows in other tables.
- Left Join: selects rows from one table that may or may not have the corresponding rows in other tables.
- Cross Join: produces a Cartesian product of the rows in two or more tables.

To join A table to B table

```
SELECT *
FROM A NATURE [INNER, LEFT, RIGHT] JOIN B
ON A.key_col = B.key_col;
```

- First, you specify the column in both tables from which you want to select data in the SELECT clause
- Second, you specify the main table i.e., A in the FROM clause.
- Third, you specify the table that the main table joins to i.e., B in the INNER JOIN clause. In addition, you put a join condition after the ON keyword i.e, A.pka = B.fka.

Natural Join

- A natural join is a join that creates an implicit join based on the same column names in the joined tables.
- A natural join can be an inner join, left join, or right join.
- If you do not specify a join explicitly e.g., INNER JOIN, LEFT JOIN, RIGHT JOIN, PostgreSQL will use the INNER JOIN by default.

```
SELECT A.col1, A.col2, B.col1, B.col2...
FROM A NATURE[INNER, LEFT, RIGHT] JOIN B;
```

The convenience

Not require you to specify the join clause because it uses an implicit join clause based on the common column

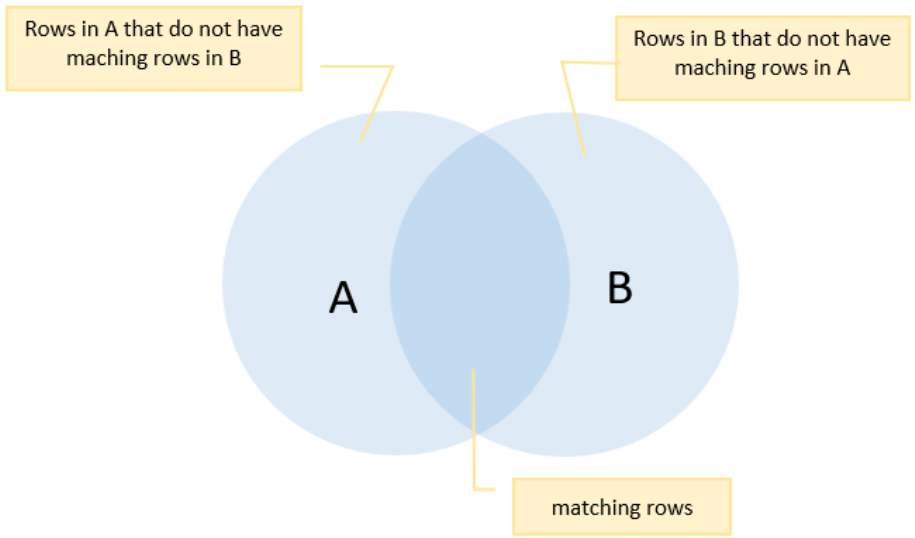
The problem

However, you should avoid using the NATURAL JOIN whenever possible because sometimes it may cause an unexpected result.

Full Outer Join

- Uses the full join to find a row in a table that does not have a matching row in another table.
- If the rows in the joined table do not match, NULL values are set for every column of the table that lacks a matching row.
- For the matching rows , a single row is included.

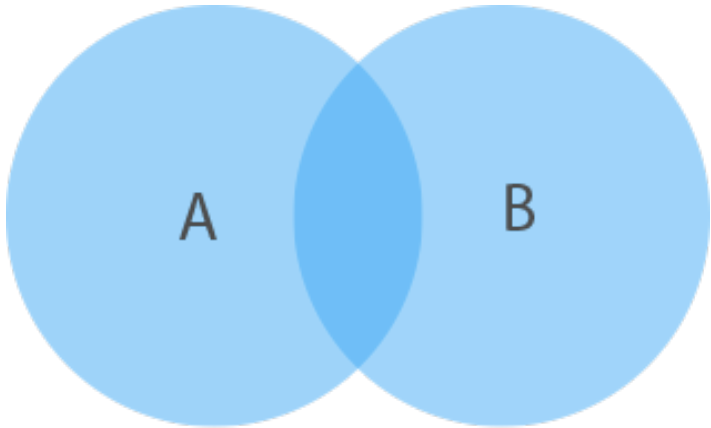
```
SELECT A.col1, A.col2, B.col1, B.col2...
FROM A [OUTER] JOIN B
ON A.key_col = B.key_col;
```



Inner Join

- Selects rows from one table that have the corresponding rows in other tables.

```
SELECT A.col1, A.col2, B.col1, B.col2...
FROM A
INNER JOIN B
ON A.key_col = B.key_col;
```

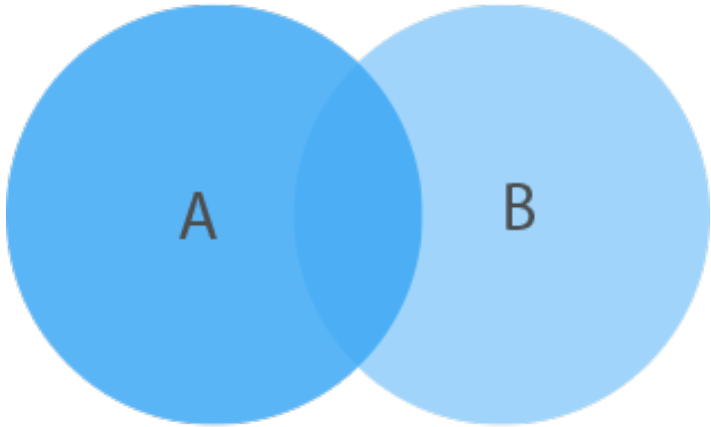


PostgreSQL INNER JOIN

Left Join

- Selects rows from one table that may or may not have the corresponding rows in other tables.

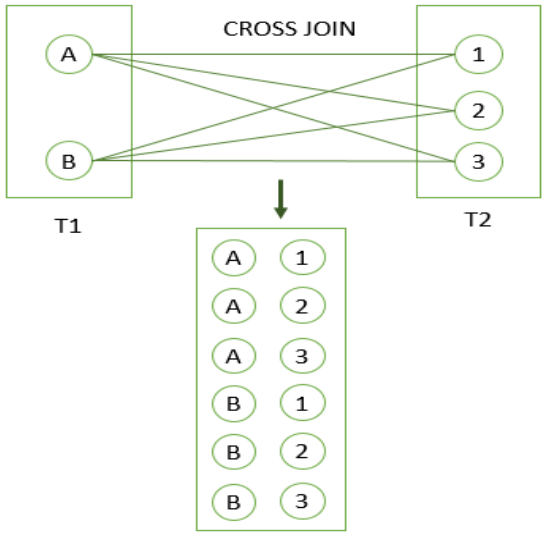
```
SELECT A.col1, A.col2, B.col1, B.col2...
FROM A
LEFT JOIN B
ON A.key_col = B.key_col;
```



PostgreSQL LEFT JOIN

Cross Join

- Produces a Cartesian product of the rows, and return every combination of rows from two tables
- Different from the other JOIN operators such as LEFT JOIN or INNER JOIN, the CROSS JOIN does not have any matching condition in the join clause.



Cross Join

```
SELECT A.col1, A.col2, B.col1, B.col2...
FROM A
CROSS JOIN B;
```

```
SELECT A.col1, A.col2, B.col1, B.col2...
FROM A, B
```

```
SELECT A.col1, A.col2, B.col1, B.col2...
FROM A
INNER JOIN B ON TRUE;
```

Set Operations Overview

- Union: combines result sets of multiple queries into a single result set.
- Intersect: combines the result sets of two or more queries and returns a single result set that has the rows appear in both result sets.
- Except: returns the rows in the first query that does not appear in the output of the second query.

To use the operators, you must obey the following rules

- The number of columns and their orders must be the same in the two queries.
- The data types of the respective columns must be compatible.

Union

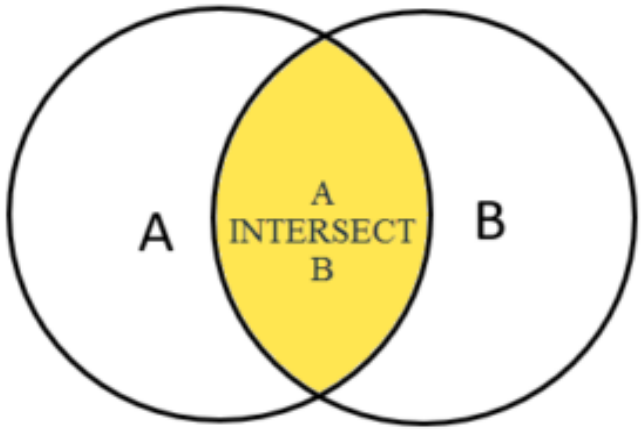
- Combines result sets of two or more SELECT statements
- The UNION operator removes all duplicate rows unless the UNION ALL is used

```
SELECT col1, col2 FROM t1
UNION
SELECT col1, col2 FROM t2;
```

Intersect

- Combines the result sets of two or more SELECT statements into a single result set, then returns all rows in the both result sets.

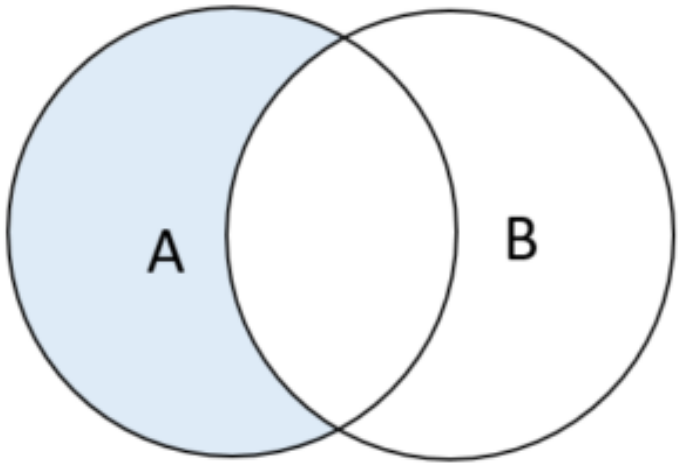
```
SELECT col1, col2 FROM t1
INTERSECT
SELECT col1, col2 FROM t2;
```



Except

- Returns distinct rows from the first (left) query that are not in the output of the second (right) query.

```
SELECT col1, col2 FROM t1 WHERE condition
EXCEPT
SELECT col1, col2 FROM t2 WHERE condition;
```



Subquery

Why subquery

Suppose we want to find the people whose grade rate is higher than the average grade in class. We can do it in two steps:

- Find the average grade by using the SELECT statement and average function.

```
SELECT AVG(grade) as avg_G FROM class;
```

  Result: avg_G=83

- Use the average grade in the second SELECT statement to find people that we want

```
SELECT student_name, grade FROM class
WHERE grade>83;
```

Can we get result in one query? The solution is to use a subquery.

Subquery

How to use subquery

A subquery is a query nested inside another query such as SELECT, INSERT, DELETE and UPDATE

```
SELECT student_name, grade FROM class
WHERE grade>(SELECT AVG(grade) FROM class);
```

Subquery with EXISTS operator

A subquery can be an input of the EXISTS operator. If the subquery returns any row, the EXISTS operator returns true. If the subquery returns no row, the result of EXISTS operator is false.

```
EXISTS subquery;
```

Useful Built in Functions

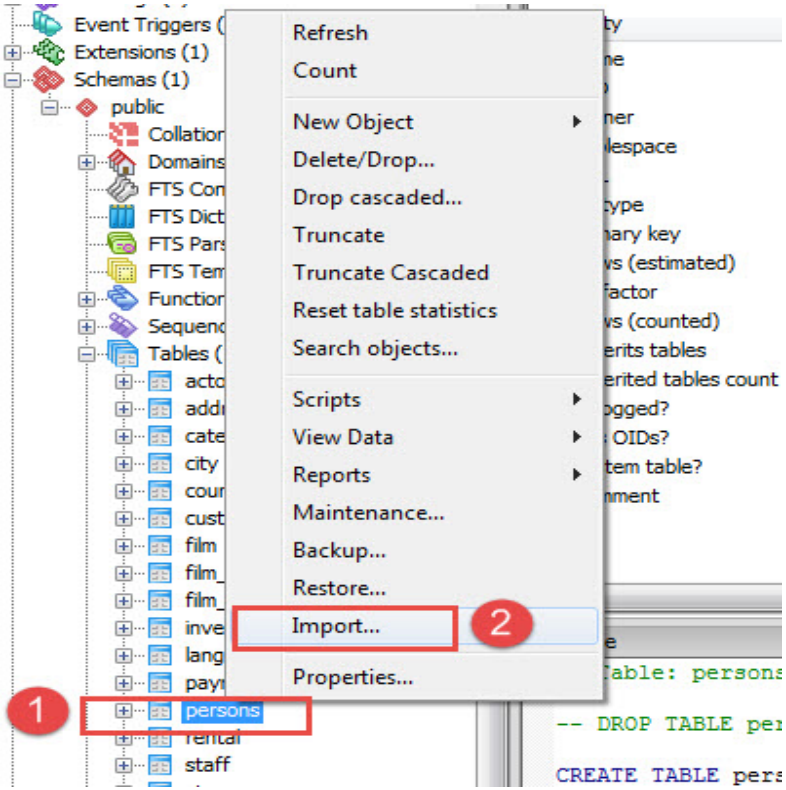
- COUNT: count the number of rows in a database table.
- MAX: select the highest (maximum) value for a certain column.
- MIN: select the lowest (minimum) value for a certain column.
- AVG: selects the average value for certain table column.
- SUM: aggregate function allows selecting the total for a numeric column.
- Numeric: a complete list to manipulate numbers in SQL.
- String: a complete list to manipulate strings in PostgreSQL.

Import from CSV file

- Using query

```
COPY table_name(col_name)
FROM 'path/filename.csv'
DELIMITER ',' CSV [HEADER];
```

- Using pgAdmin



Export to CSV file

- Using query

```
COPY table_name(col_name)
TO 'path/filename.csv'
DELIMITER ',' CSV [HEADER];
```

- Using \copy
  - In case you have the access to a remote PostgreSQL database server, but you don't have sufficient privileges to write to a file on it, you can use the PostgreSQL built-in command \copy.