Can visual diagrams help users understand existing SQL queries?

- In this study, we verify whether **visual diagrams** can help users recognize and understand the intent of a **SQL query** faster and more accurately than SQL by itself.

- The goal of the following 9 pages is to provide you with a quick introduction to our study setup on AMT, an overview of the database schema used for all the 12 queries in the study, and a tutorial on how to read the visual diagrams.

- For each of the 12 questions in the study you will see either a **SQL query**, or a **visual diagram**, or **both.**

- Your task is to choose the best interpretation of the query among the 4 choices.

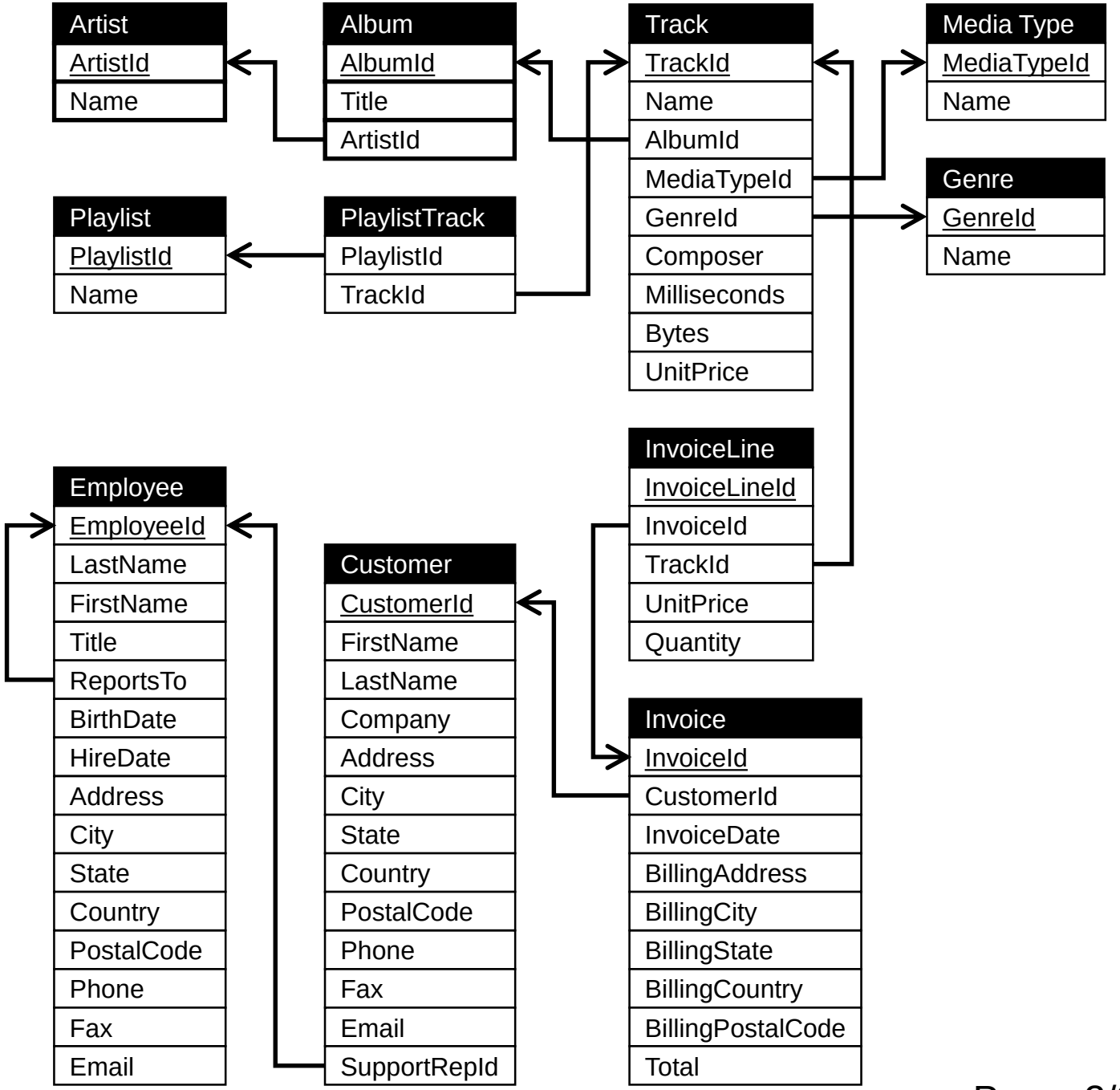
- Proceed to the next page to see the database schema you will be using and sample questions.

- Use the buttons below or the keyboard's left and right keys to navigate through the tutorial.

All questions during the test will be using the relational schema of a music publisher's digital media store, including tables for artists, albums, media tracks, invoices and customers.

On the right, primary keys are underlined, and foreign keys point to the primary key they refer to.

- A track is from one album.
- Each album is by one artist.
- A track has one media type (mp3, AAC, etc.)
- A track has one genre (pop, rock, rap, etc.)
- A purchased track has an invoice line.
- An invoice is created each time a customer makes a purchase.
- An invoice has one or more invoice lines, one for each track purchased.
- The invoice line "quantity" attribute records how many copies of the same track were purchased in one invoice. Thus customers can purchase the same track multiple times!
- Attribute "Milliseconds" records the duration/length of a track.
- Each customer has an employee support representative.
- Each employee reports to another employee.



Basic Conjunctive Query

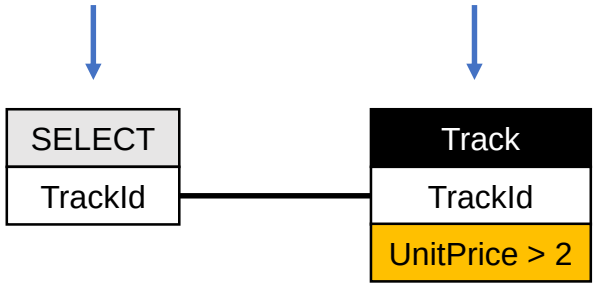
```
SELECT    T.TrackId
FROM      Track T
WHERE     T.UnitPrice > 2;
```

The visual diagram for a query is read as follows:

(1) Select the attribute TrackId …

(2)… from the table Track…

(3)… where unit price is greater than 2.



The diagrams show predicates like "T.UnitPrice > 2" with a yellow background.

Interpretation:
Find TrackId of Tracks whose UnitPrice is greater than 2.

Basic Query With Joins

```
SELECT    T.TrackId
FROM      Track T, PlaylistTrack PT, Playlist P, Genre G
WHERE     T.GenreId = G.GenreId
AND       T.TrackId = PT.TrackId
AND       PT.PlaylistId = P.PlaylistId
AND       G.Name <> P.Name;
```

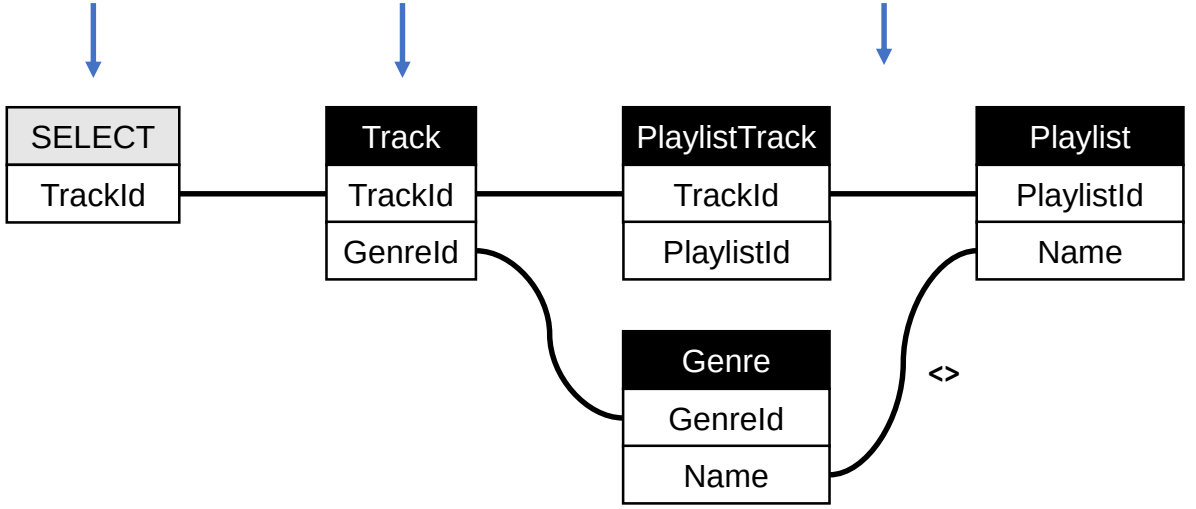
The visual diagram for a query is read as follows:

(1) Select the attribute TrackId …

(2)… from the table Track where the Track …

(3)… is in a playlist…

(4)… whose name is different from the track's genre.



- *Cross-table joins are represented by a line connecting the joining attributes from the two tables.*
- *An unlabeled line represent an equijoin* (=).
- *A labeled line represents a join applying the logic operator of its label. In the example above, the line between G.Name and P.Name* is labeled with **<>**, indicating that we join using the 'not equals' operator.

Interpretation:
"Find the TrackId of Tracks that are in some Playlist whose name is different from the Genre of the Track."

Group By Queries with Aggregates

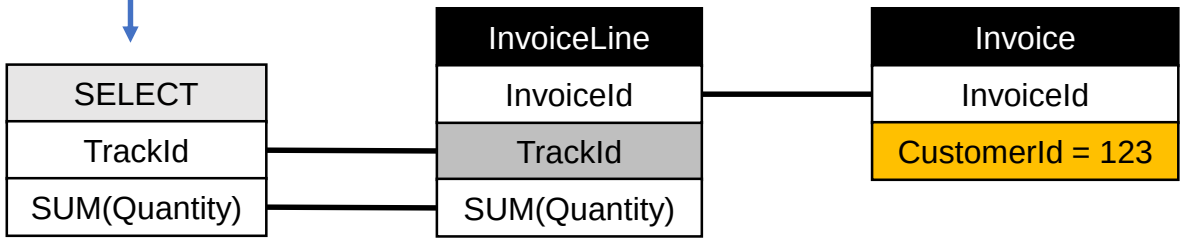
```
SELECT      T.TrackId, SUM(IL.Quantity)
FROM        InvoiceLine IL, Invoice I
WHERE       IL.InvoiceId = I.InvoiceId
AND         I.CustomerId = 123
GROUP BY    T.TrackId;
```

The visual diagram for a query is read as follows:

(1) Select the attributes TrackId and SUM(Quantity)…

*(2)… from table InvoiceLine **grouped by** TrackId and **summed up** Quantities …*

(3)… for invoices from the customer with ID 123.



- *The attribute that we "**Group By**" is highlighted with a grey background.*

Interpretation:
"For each TrackId find the total sale quantity bought by the customer with ID = 123."

Basic Nested (Not Exist Query)

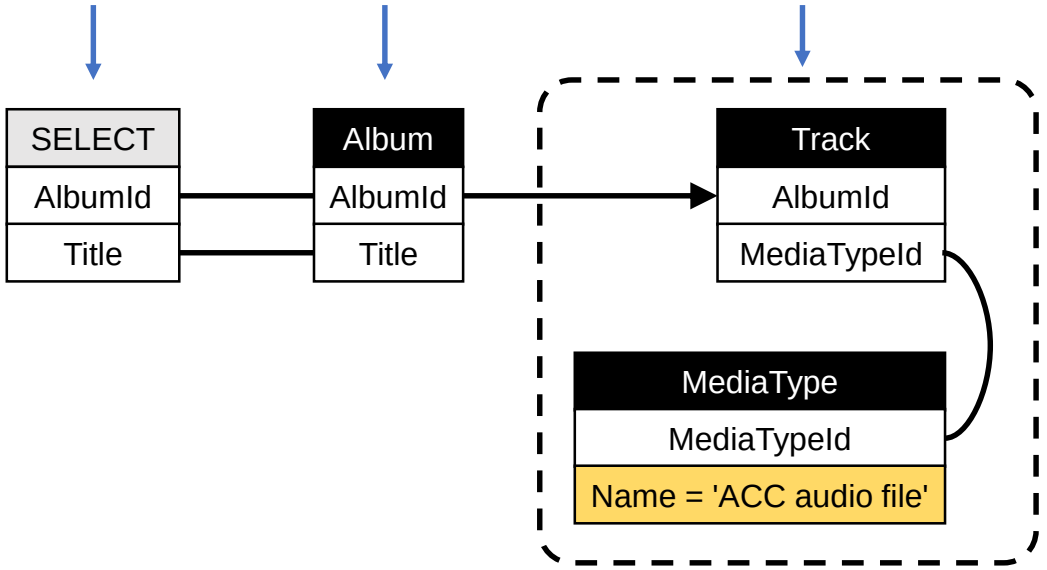
```
SELECT  AL.AlbumId, AL.Title
FROM    Album AL
WHERE   NOT EXISTS
        (SELECT *
        FROM    Track T, MediaType MT
        WHERE   AL.AlbumId = T.AlbumId
        AND     T.MediaTypeId = MT.MediaTypeId
        AND     MT.Name = 'ACC audio file');
```

The visual diagram for a query is read as follows:

(1) Select the attributes AlbumId and Title…

(2)… from the table Album …

*(3)… for which there **does not exist any** track whose MediaType name is 'ACC audio file'.*



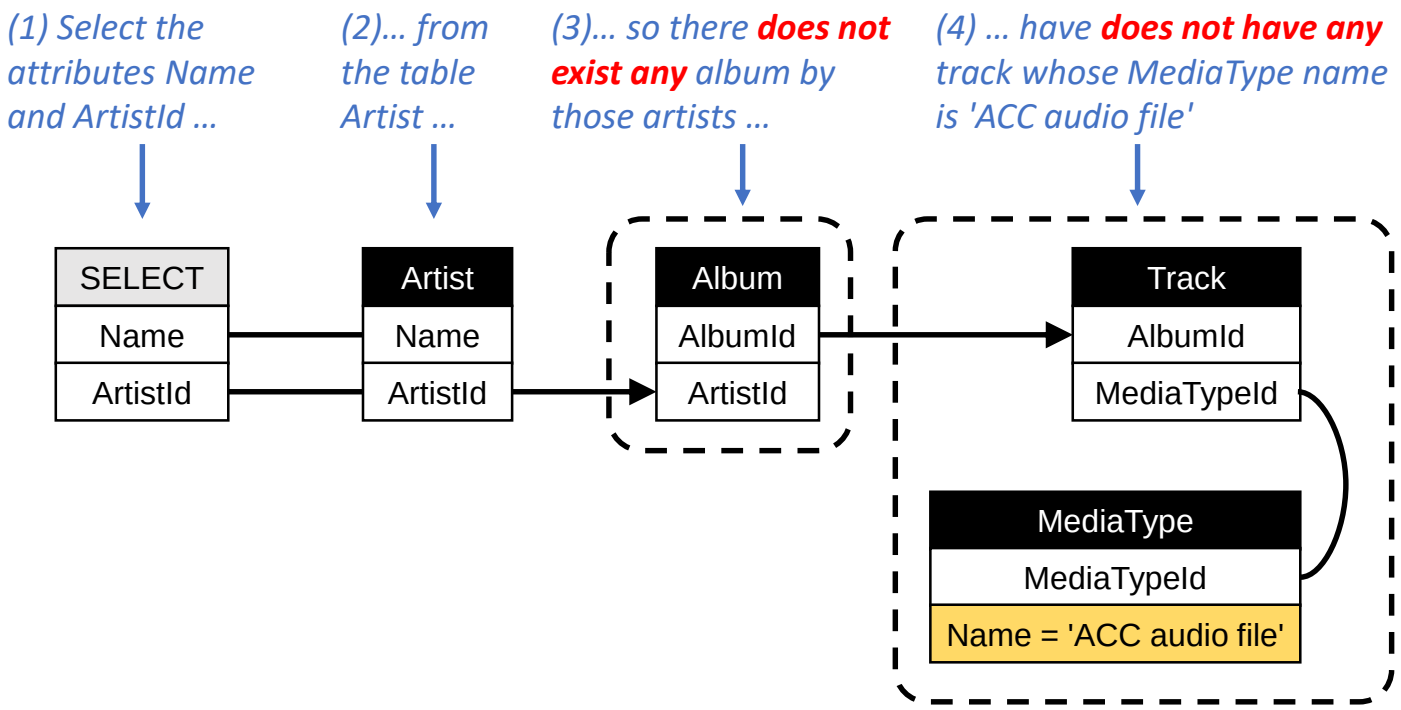
- *Boxes with **dashed lines** represent logical "**not exists**" relationships.*
- *The arrows point along the reading order into a dashed box.*

Interpretation:
"Find AlbumId and Title of Albums for which no Track is available as 'ACC audio file' MediaType."

Example double nested SQL query

```
SELECT  A.Name, A.ArtistId
FROM    Artist A
WHERE   NOT EXISTS
        (SELECT *
        FROM    Album AL
        WHERE   AL.ArtistId = A.ArtistId
        AND     NOT EXISTS
                (SELECT *
                FROM    Track T, MediaType MT
                WHERE   AL.AlbumId = T.AlbumId
                AND     T.MediaTypeId = MT.MediaTypeId
                AND     MT.Name = 'ACC audio file')
        );
```

The visual diagram for a query is read as follows:



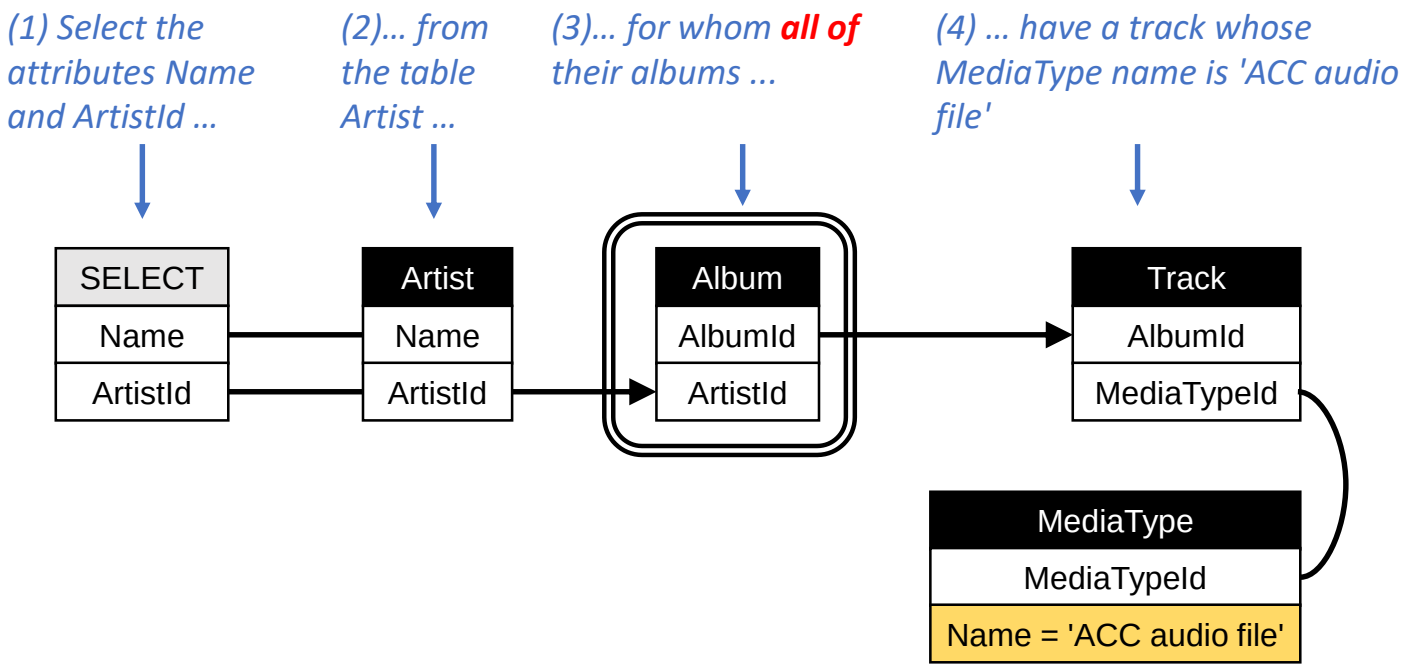
- *Boxes with **dashed lines** represent logical "**not exists**" relationships.*
- *SQL cannot express "for all statements (e.g., "so that all entities have an attribute") and thus need express those by double negation (e.g., "so that no entities does not have an attribute").*
- *The arrows point along the reading order into a dashed box.*

Interpretation:
"Find Name and ArtistId of Artists who have no Album that does not have any Track whose MediaType name is 'ACC audio file'."
or
"Find Name and ArtistId of Artists for whom all their Albums contain at least one Track whose MediaType name is 'ACC audio file'."

Example double nested SQL query ("for all" simplification)

```
SELECT  A.Name, A.ArtistId
FROM    Artist A
WHERE   NOT EXISTS
        (SELECT *
        FROM    Album AL
        WHERE   AL.ArtistId = A.ArtistId
        AND     NOT EXISTS
                (SELECT *
                FROM    Track T, MediaType MT
                WHERE   AL.AlbumId = T.AlbumId
                AND     T.MediaTypeId = MT.MediaTypeId
                AND     MT.Name = 'ACC audio file')
        );
```

The visual diagram for a query is read as follows:



- *Boxes with **double lines** represent logical "**for all**" relationships.*
- *Thus in contrast to SQL, the visual diagrams can express **for all statements** and can thus avoid double negation.*
- *The arrows point along the reading order into and out of a double lined box.*

Interpretation:

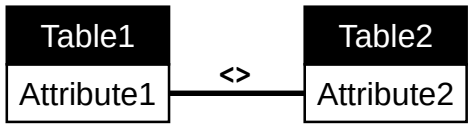
"Find Name and ArtistId of Artists who have no Album that does not have any Track whose MediaType name is 'ACC audio file'."

or

"Find Name and ArtistId of Artists for whom all their Albums contain at least one Track whose MediaType name is 'ACC audio file'."

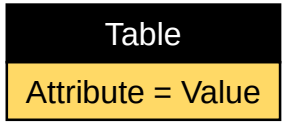
Legend for visual diagrams

Joins with an inequality predicate are shown with a line and label.



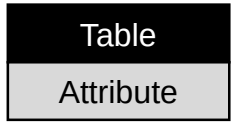

```
WHERE  Table1.Attribute1 <> Table2.Attribute2
```

Boxes with a yellow background show **selection predicates** on that Attribute and the value being matched.



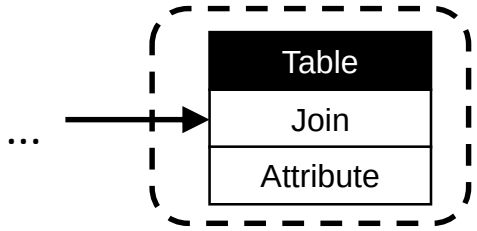

```
WHERE  Table.Attribute = Value
```

Boxes with a gray background show **Group By** operations on that Attribute.



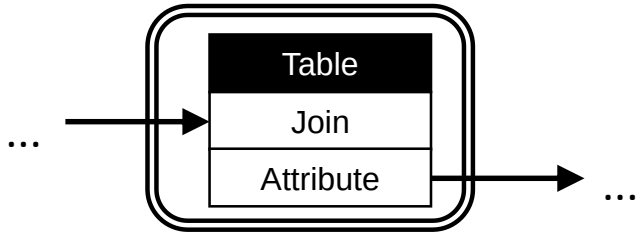

```
GROUP BY  Table.Attribute
```

Logical not exists relations are shown using a dashed line around the affected tables.



"... so there does not exist any Table with Attribute."

Logical for all relations are shown using two solid lines around the affected tables.



"... so that for all tables with Attribute, ..."

You can always go back to the tutorial and this summary legend while answering the 12 queries.

To do that, click at the Tutorial PDF Link at the bottom banner of the test.