How Many Ways Can You Join SAS® Tables

Part 2

Charu Shankar
SAS Education

Bio

With a background in computer systems management. SAS Instructor Charu Shankar engages with logic, visuals, and analogies to spark critical thinking.

A SAS Instructor since 2007, she curates and delivers unique content via the SAS YouTube channel, SAS global forum, SAS Ask the Expert Series, SAS Training Post Blog, etc.

Charu loves to support users by teaching at conferences on topics related to SAS, SQL, Efficiencies, PERL, Macros, Python, Viya, etc.

When she's not coding, Charu is A Yoga Instructor who loves to explore Canadian trails with her husky Miko.

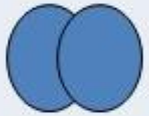
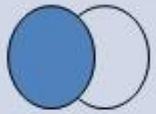
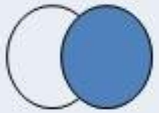
Agenda

- Nuts and Bolts
- PROC SQL Join
- PROC SQL Inline View
- PROC SQL Join with Implicit and Explicit Pass-Through
- Handy Links

Nuts and Bolts

Why Join Anyway

Merging SQL Vs. SAS Data Step

VISUAL	SQL	DATA STEP
All rows from both tables 	**Full Outer Join** Select * from tableA Full outer Join tableB On tableA.id=tableB.id;	**Match Merge** Data tableC; Merge tableA tableB; By id; Run;
All rows from left table & matching rows from right table 	**Left Join** Select * from tableA Left Join tableB On tableA.id=tableB.id;	**Data Step Merge** Use IN=data set option Data tableD; Merge tableA(in=INA) tableB; By id; If INA; Run;
All rows from right table & matching rows from left table 	**Right Join** Select * from tableA Right Join tableB On tableA.id=tableB.id;	**Data Step Merge** Data tableE; Merge tableA tableB(in=INB); By id; If INB; Run;

What happens during a Join?

2 phases

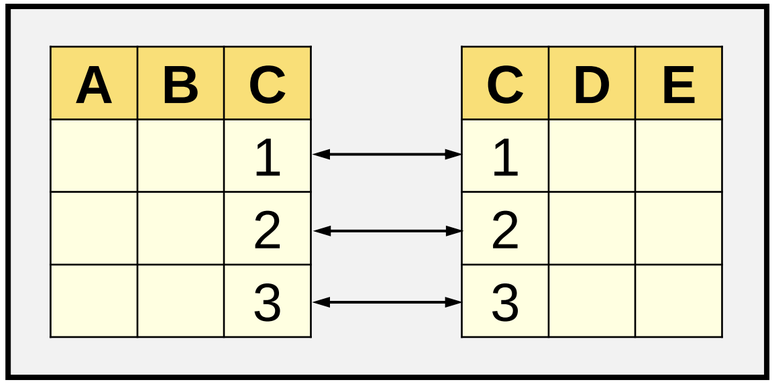
1. The first phase determines the names of the tables referenced in the FROM clause. An internal *virtual* table, known as a Cartesian product, is created resulting in each row in the first table being combined with each row in the second table, and so forth. Due to its size, the Cartesian product is managed by the SAS software.

2. The second phase of every join processes the WHERE clause, when present.

Types of Joins in PROC SQL

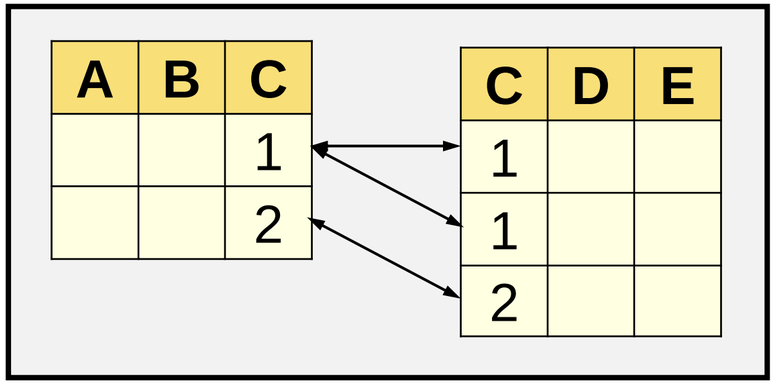
Type of Join	Quality of Join
Inner Join	Returns matching rows in 2 tables
Outer Join	Returns matches/non matches from 2 tables
Cartesian (Cross-Join)	Returns the Cartesian product of rows from the tables in the join, all possible combination of rows are returned
Self Join (Reflexive Join)	Join that joins a table back with itself
Natural Join	JOIN operation that creates an implicit join clause based on the common columns in the tables being joined.
Inline View	SELECT statement in the FROM-clause of another SELECT statement to create a temporary table that could be referenced by the SELECT statement. Inline views are utilized for writing complex SQL queries without join and subqueries operations.
Subquery	Inner Query that sits inside an Outer Query

Data Relationships & Cardinality



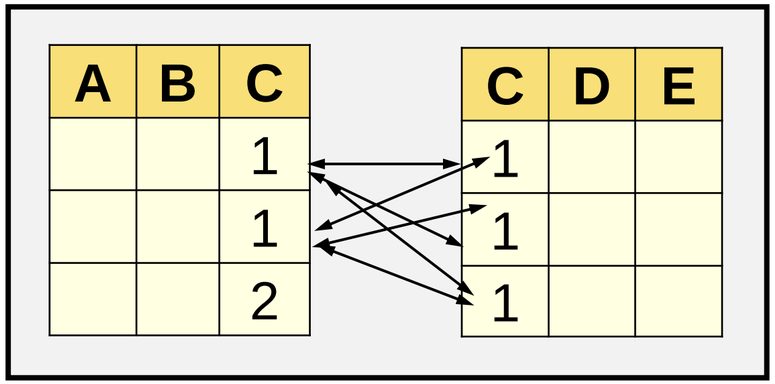
One-to-One

Each row in one table is linked (or related) to a single row in another table using a “key” column.



One-to-Many

Each row in one table is linked (or related) to one, or more, rows in another table using a “key” column.



Many-to-Many

One, or more, rows in one table is linked (or related) to one, or more, rows in another table using a “key”

PROC SQL Joins

Inner, Outer

SELECT Statement: Syntax Order Refresher

SO FEW WORKERS GO HOME ON TIME	**SELECT** *object-item <, ...object-item>* **FROM** *from-list* <**WHERE** *sql-expression*> <**GROUP BY** *object-item <, ... object-item >>* <**HAVING** *sql-expression*> <**ORDER BY** *order-by-item <DESC>* *<, ...order-by-item>>*;

- The WHERE clause specifies data that meets certain conditions.
- The GROUP BY clause groups data for processing.
- The HAVING clause specifies groups that meet certain conditions.
- The ORDER BY clause specifies an order for the data.

SELECT Statement: Required Clauses

SELECT *object-item <, ...object-item>*
FROM *from-list;*

Here are two things that SQL always needs:

1. What do you want?
   The SELECT clause specifies the columns and column order.
2. Where do you want it from?
   The FROM clause specifies the data sources.
   You can query from 1 to 256 tables.

Joins and a Cartesian Product

Input Tables

Authors
AuthorID
AuthorName

Books
BookID
BookTitle
AuthorID

Joins and a Cartesian Product

PROC SQL Join Query - Syntax

```
PROC SQL;
SELECT *
    FROM SAS.Authors,
         SAS.Books(keep=BookID BookTitle
                                AuthorID);
QUIT;
```

Joins and a Cartesian Product

Non Matching Author Id

PROC SQL Join - Results

AuthorID	AuthorName	AuthorBio	BookID	BookTitle	AuthorID
A001	Tricia Aanderud		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A002	Robert Allison		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
B001	William Benjamin		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
B002	Jonas V. Bilenas		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
B003	Michele M. Burlew		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
C001	Art Carpenter		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
C002	Goutam Chakraborty		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
C003	Ron Cody		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
D001	Lora D. Delwiche		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
D002	Barry de Ville		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
D003	Craig Dickstein		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
D004	Paul Dorfman		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
E001	Peter Eberhardt		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
E002	Jane Eslinger		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
F001	Lisa Fine		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
G001	Sunil K. Gupta		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H001	Angela Hall		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H002	Lauren Haworth		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H003	Dan Heath		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H004	Chris Hemedinger		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H005	Don Henderson		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H006	Philip Holland		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
J001	Mark Jordan		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
K001	Warren F. Kuhfeld		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
L001	Kirk Paul Lafler		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001

Exponential # of rows

Debugging SQL Processing _METHOD

```
80   PROC SQL _method;
81   SELECT *
82   FROM SAS.Authors,
83   SAS.Books(keep=BookID BookTitle AuthorID);
NOTE: The execution of this query involves performing one or more Cartesian product joins that can not be optimized.
NOTE: SQL execution methods chosen are:
      sqxslct
          sqxjsl
              sqxsrc( SAS.AUTHORS )
              sqxsrc( SAS.BOOKS )
84   QUIT;
```

CODES	DESCRIPTION
Sqxcrta	Create table as Select
Sqxslct	Select
Sqxjsl	Step loop join (Cartesian)
Sqxjm	Merge join
Sqxjndx	Index join
Sqxjhsh	Hash join
Sqxsort	Sort
Sqxsrc	Source rows from table
Sqxfil	Filter rows
Sqxsumg	Summary stats with GROUP BY
Sqxsumn	Summary stats with no GROUP BY

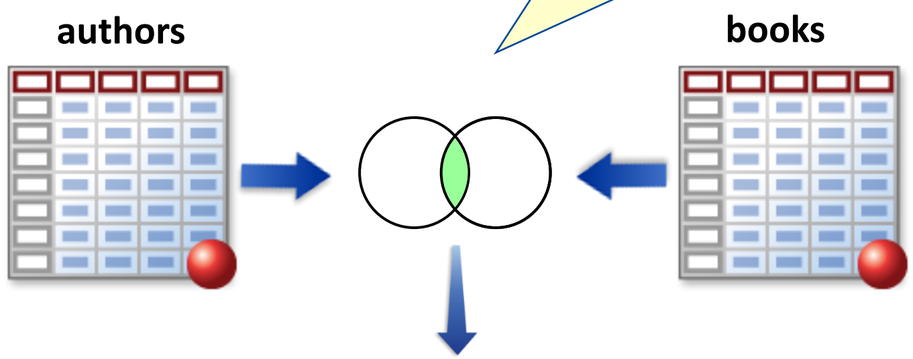
Debugging SQL Processing _TREE

```
NOTE: The execution of this query involves performing one or more Cartesian product joins that can not be optimized.
Tree as planned.
                                /-SYM-V-(Authors.AuthorId:1 flag=00000001)
                     /-OBJ----|
                    |         |--SYM-V-(Authors.AuthorName:2 flag=00000001)
                    |         |--SYM-V-(Authors.AuthorBio:3 flag=00000001)
                    |         |--SYM-V-(Books.BookID:1 flag=00000001)
                    |         |--SYM-V-(Books.BookTitle:2 flag=00000001)
                    |          \-SYM-V-(Books.AuthorID:3 flag=00000001)
          /-JOIN---|
         |          |                              /-SYM-V-(Authors.AuthorId:1 flag=00040001)
         |          |                     /-OBJ----|
         |          |                    |         |--SYM-V-(Authors.AuthorName:2 flag=00040001)
         |          |                    |          \-SYM-V-(Authors.AuthorBio:3 flag=00040001)
         |          |          /-SRC----|
         |          |         |          \-TABL[SAS].Authors opt=''
         |          \-FROM---|
         |                    |                     /-SYM-V-(Books.BookID:1 flag=00040001)
         |                    |          /-OBJ----|
         |                    |         |         |--SYM-V-(Books.BookTitle:2 flag=00040001)
         |                    |         |          \-SYM-V-(Books.AuthorID:3 flag=00040001)
         |                     \-SRC----|
         |                              \-TABL[SAS].Books opt='keep=BookID BookTitle AuthorID'
 --SSEL---|
```

Conventional Joins with 2 Tables and a Where Clause

PROC SQL Inner Join - Concepts

A conventional join of 2 or more tables, uses a WHERE- or HAVING-clause to produce a result set of “matched” rows. The Authors and Books table are joined together using the “key” AuthorID.



AuthorId	AuthorName	BookId	BookTitle	AuthorID
S008	Susan Slaughter	S00801	The Little SAS Book: A Primer, Fifth Edition	S008

Conventional Joins with 2 Tables and a Where Clause

PROC SQL Inner Join - Syntax

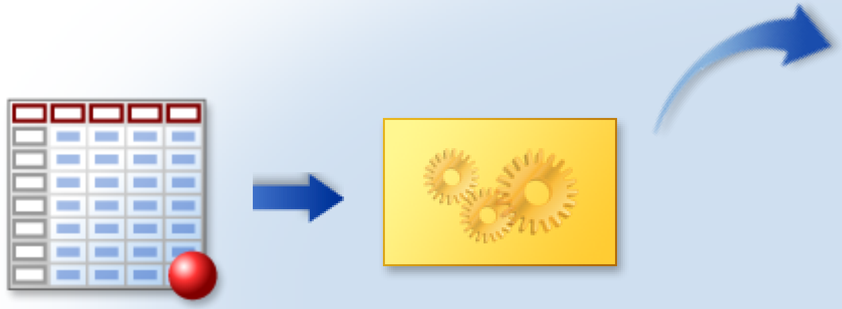
```
PROC SQL;
SELECT *
    FROM SAS.Authors
    INNER JOIN
    SAS.Books(keep=BookID BookTitle
                                AuthorID)
    ON Authors.AuthorID = Books.AuthorID;
QUIT;
```

```
PROC SQL;
SELECT *
     FROM t1
     INNER JOIN
     t2
     ON t1.key=t2.key;
QUIT;
```

"key", AuthorID in a WHERE-clause. When the value of AuthorID is equal in both tables, the rows are combined together.

Locate Key Columns For Joins

How did I know what key columns the 2 tables Authors & Books in common?



Column Name	Member Name	Column Type	Column Length
AuthorId	AUTHORS	char	8
AuthorID	BOOKS	char	8

Locate Common Columns For Joins

```
proc sql;
    describe table dictionary.columns;
NOTE: SQL table DICTIONARY.COLUMNS was created like:
create table DICTIONARY.COLUMNS
  (
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   memtype char(8) label='Member Type',
   name char(32) label='Column Name',
   type char(4) label='Column Type',
   length num label='Column Length',
   npos num label='Column Position',
   varnum num label='Column Number in Table',
   label char(256) label='Column Label',
   format char(49) label='Column Format',
   informat char(49) label='Column Informat',
   idxusage char(9) label='Column Index Type',
   sortedby num label='Order in Key Sequence',
   xtype char(12) label='Extended Type',
   notnull char(3) label='Not NULL?',
   precision num label='Precision',
   scale num label='Scale',
   transcode char(3) label='Transcoded?'
  );
```

Know your dictionary table

Locate Common Columns For Joins

How can I search without hard coding?

```
proc sql;
  select name, memname, type, length from dictionary.columns
    where libname ='SAS'
      group by upcase(name)
        having count(upcase(name)) > 1
          order by upcase(name);
quit;
```

Column Name	Member Name	Column Type	Column Length
AuthorId	AUTHORS	char	8
AuthorID	BOOKS	char	8

Conventional Joins with 2 Tables and a Where Clause

PROC SQL Inner Join - Syntax

```
PROC SQL;
SELECT *
    FROM SAS.Authors
    INNER JOIN
    SAS.Books(keep=BookID BookTitle
                                AuthorID)
    ON Authors.AuthorID = Books.AuthorID;
QUIT;
```

"key", AuthorID in a WHERE-clause. When the value of AuthorID is equal in both tables, the rows are combined together.

Conventional Joins with 2 Tables and a Where Clause

PROC SQL Inner Join - Results

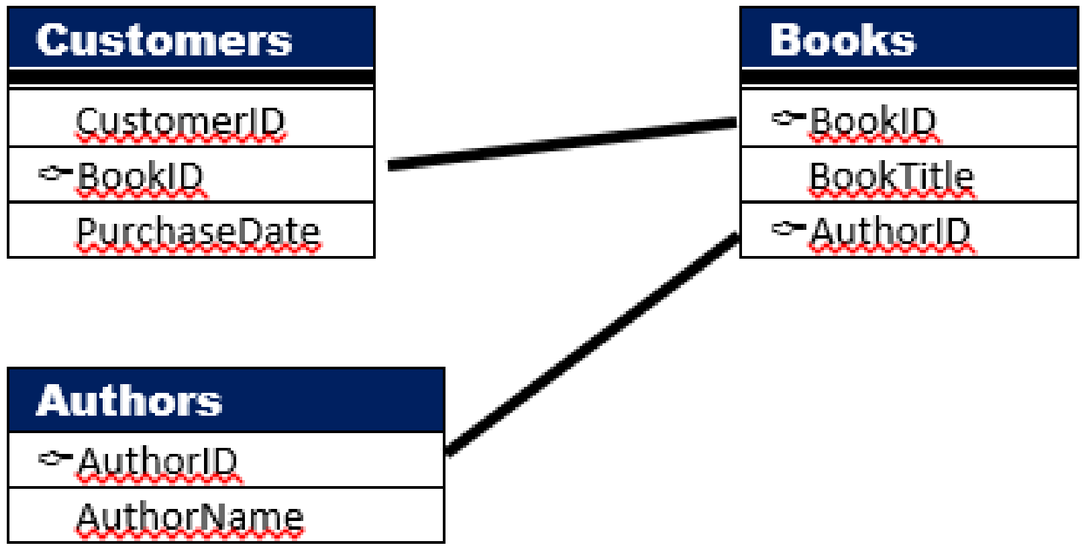
AuthorID	AuthorName	BookID	BookTitle	AuthorID
A001	Tricia Aanderud	A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A001	Tricia Aanderud	A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight Into Your Data	A001
B003	Michele M. Burlew	B00304	SAS® Macro Programming Made Easy, Third Edition	B003
B003	Michele M. Burlew	B00303	SAS® Hash Object Programming Made Easy	B003
B003	Michele M. Burlew	B00302	Combining and Modifying SAS® Data Sets: Examples, Second Edition	B003
B003	Michele M. Burlew	L00201	Output Delivery System: The Basics and Beyond	B003
C001	Art Carpenter	C00101	Carpenter's Complete Guide to the SAS® REPORT Procedure	C001
C001	Art Carpenter	C00102	Carpenter's Guide to Innovative SAS® Techniques	C001
C001	Art Carpenter	C00103	Carpenter's Complete Guide to the SAS® Macro Language, Third Edition	C001
C003	Ron Cody	C00309	Cody's Data Cleaning Techniques Using SAS®, Third Edition	C003
C003	Ron Cody	C00308	Biostatistics by Example Using SAS® Studio	C003
C003	Ron Cody	C00307	An Introduction to SAS® University Edition	C003
C003	Ron Cody	C00306	Test Scoring and Analysis Using SAS®	C003
C003	Ron Cody	C00305	Cody's Collection of Popular SAS® Programming Tasks a	C003

AuthorID column is displayed twice in the results – once from the Authors table and a second time from the Books table.

Conventional Joins with 3 Tables and a Where Clause

Joins with 3 tables - Concepts

A join with 3 tables adheres to the same rules as a 2-table join. Each table needs to be listed in the FROM clause with the appropriate subsetting (or matching) restrictions specified in a WHERE clause.



Conventional Joins with 3 Tables and a Where Clause

Joins with 3 tables - Syntax

```
PROC SQL;
    SELECT A.AuthorID, AuthorName, B.BookID, BookTitle, PurchaseDate
      FROM  SAS.Authors(drop=AuthorBio) AS A
        INNER JOIN
            SAS.Books(keep=BookID BookTitle AuthorID) AS B
            ON A.AuthorID = B.AuthorID
        INNER JOIN
            SAS.Customers1 AS C
            ON B.BookID = C.BookID;
QUIT;
```

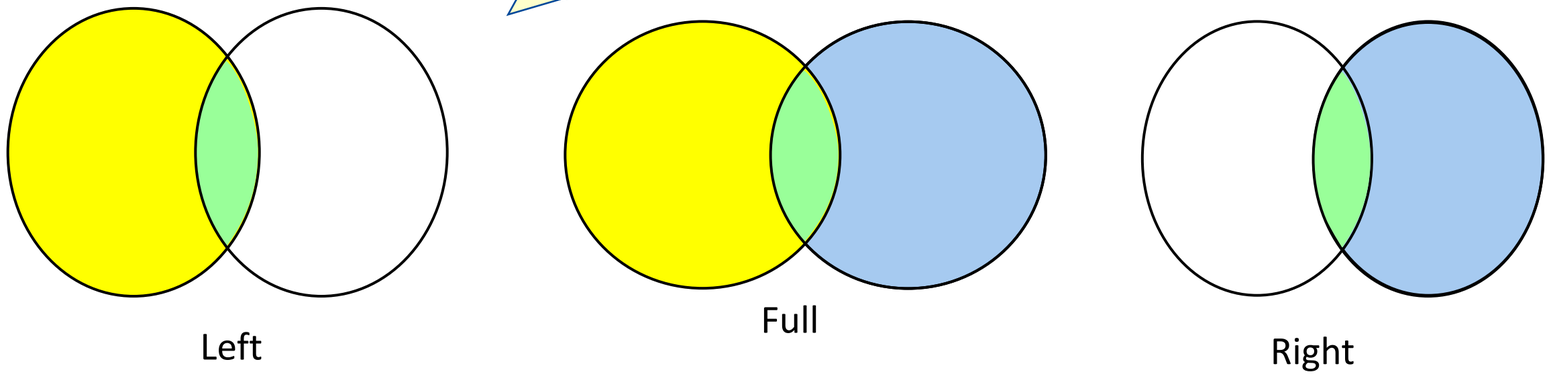
Conventional Joins with 3 Tables and a Where Clause

Joins with 3 tables – Results

AuthorID	AuthorName	BookID	BookTitle	PurchaseDate
A001	Tricia Aanderud	A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight into Your Data	05/11/2017
B003	Michele M. Burlew	B00304	SAS® Macro Programming Made Easy, Third Edition	02/28/2015
B003	Michele M. Burlew	L00201	Output Delivery System: The Basics and Beyond	11/03/2015
C001	Art Carpenter	C00103	Carpenter's Complete Guide to the SAS® Macro Language, Third Edition	02/28/2015
C003	Ron Cody	C00309	Cody's Data Cleaning Techniques Using SAS®, Third Edition	04/18/2017
C003	Ron Cody	C00309	Cody's Data Cleaning Techniques Using SAS®, Third Edition	03/27/2016
C003	Ron Cody	C00309	Cody's Data Cleaning Techniques Using SAS®, Third Edition	11/03/2015

Outer Joins

matching rows are selected along with the unmatched rows from one, both or all tables. sometimes referred to as an asymmetric (or unconventional) join. Its basic purpose is to select the matching rows from all tables, and to capture the rows without a match row from all tables.

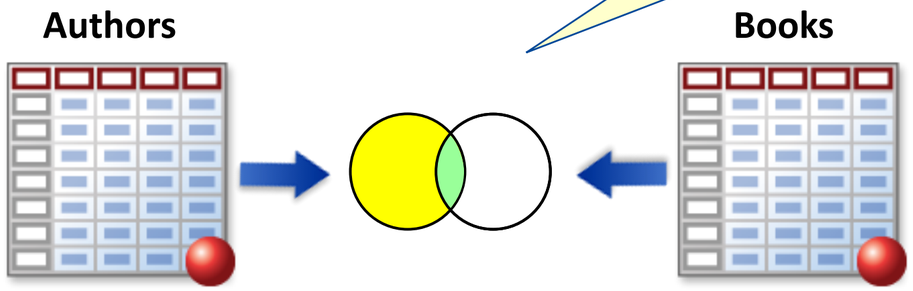


This type of join construct serves a significant purpose when working with tables of data and is referred to as an outer join construct.

Left Outer Joins

Left Outer Join - Concepts

a left outer join is constructed to select the “matched” AuthorIDs from both the Authors and Books tables, plus all the “unmatched” rows from the Authors table.



Left Outer Joins

Left Outer Join - Syntax

```
PROC SQL;
  SELECT Authors.AuthorID, BookTitle, HardcoverPrice
    format=Dollar8.2
    FROM  SAS.Authors
          LEFT JOIN
          SAS.Books
          ON Authors.AuthorID = Books.AuthorID;
QUIT;
```

Left Outer Joins

Left Outer Join – Results

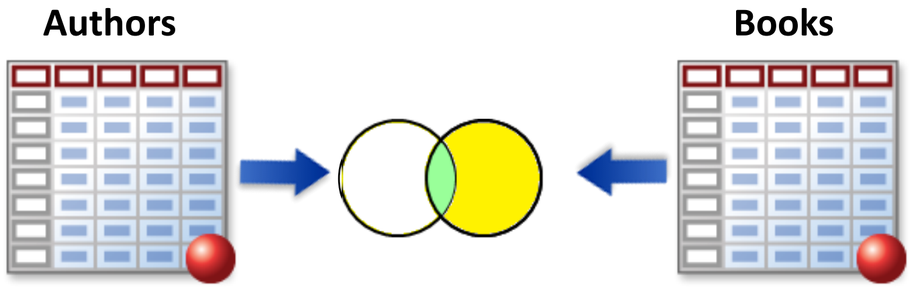
AuthorID	BookTitle	HardcoverPrice
A001	Building Business Intelligence Using SAS: Content Development Examples	$63.95
A001	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight into Your Data	$49.95
A002		.
B001		.
B002		.
B003	Output Delivery System: The Basics and Beyond	$39.98

The result contains all rows matching the rows from the left table (Authors) that did not match any row in the right (Books) table. Essentially any “unmatched” rows from the left table are preserved and displayed as they appear in the table itself.

Right Outer Joins

Right Outer Join - Concepts

An example of a right outer join to identify and match AuthorIDs from the Authors and Books tables



Right Outer Joins

Right Outer Join - Syntax

```
PROC SQL;
    SELECT Authors.AuthorID, BookTitle, HardcoverPrice
    format=Dollar8.2
        FROM SAS.Authors
            RIGHT JOIN
                SAS.Books
                ON Authors.AuthorID = Books.AuthorID;
QUIT;
```

Right Outer Joins

Right Outer Join - Results

AuthorID	BookTitle	HardcoverPrice
A001	Building Business Intelligence Using SAS: Content Development Examples	$63.95
A001	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight into Your Data	$49.95
B003	Output Delivery System: The Basics and Beyond	$39.98
B003	SAS® Hash Object Programming Made Easy	$29.95
B003	SAS® Macro Programming Made Easy, Third Edition	$59.95
B003	Combining and Modifying SAS® Data Sets: Examples, Second Edition	$48.95
C001	Carpenter's Complete Guide to the SAS® Macro Language, Third Edition	$74.95
C001	Carpenter's Complete Guide to the SAS® REPORT Procedure	$74.95

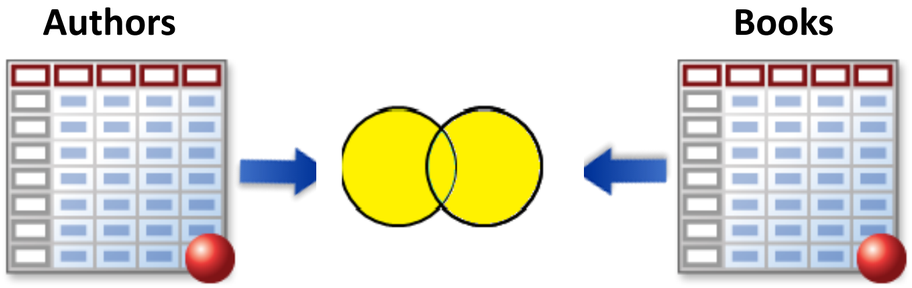
The result contains all rows for which the SQL expression, referenced in the ON clause, matches the rows from the right table (Books) that did not match any row in the left (Authors) table.

Full Outer Joins

Full Outer Join - Concepts

A full outer join essentially represents the result of a left outer join and a right outer join. The result of a full outer join can be sizeable because it contains all “matches” and “non matches” from both the left and right table.

A full outer join query is constructed that selects columns from the Authors and Books tables.



Full Outer Joins

Full Outer Join - Syntax

```
PROC SQL;
    SELECT Authors.AuthorID, BookTitle, HardcoverPrice
format=Dollar8.2
        FROM SAS.Authors
            FULL JOIN
                SAS.Books
                ON Authors.AuthorID = Books.AuthorID;
QUIT;
```

Full Outer Joins

Full Outer Join - Results

AuthorID	BookTitle	HardcoverPrice
A001	Building Business Intelligence Using SAS: Content Development Examples	$63.95
A001	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight into Your Data	$49.95
A002		.
B001		.
B002		.
B003	Output Delivery System: The Basics and Beyo	$39.98
B003	SAS® Hash Object Programming Made Easy	$29.95
B003	SAS® Macro Programming Made Easy, Third Editi	$59.95

The result contains all rows that satisfy the SQL expression, referenced in the ON clause, by matching the rows from the right table (Books) that did not match any row in the left (Authors) table.

Full Outer Joins

Full Outer Join - Results

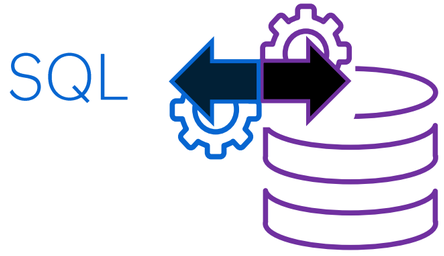
AuthorID	BookTitle	HardcoverPrice
A001	Building Business Intelligence Using SAS: Content Development Examples	$63.95
A001	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight into Your Data	$49.95
A002		.
B001		.
B002		.
B003	Output Delivery System: The Basics and Beyo	$39.98
B003	SAS® Hash Object Programming Made Easy	$29.95
B003	SAS® Macro Programming Made Easy, Third Editi	$59.95

The result contains all rows that satisfy the SQL expression, referenced in the ON clause, by matching the rows from the right table (Books) that did not match any row in the left (Authors) table.

DEMO : Inner and Outer Joins

PROC SQL Implicit & Explicit Pass Through

Implicit versus Explicit Pass-Through



Implicit Pass-Through

SAS will attempt to **convert SAS SQL** to native database SQL wherever possible.

Explicit Pass-Through

Use **native database SQL** to process the data directly in the database.

Implicit passthrough tries to hand off as much processing (including joins, aggregations etc) as it can to the database to limit the amount of movement of data to and from the database. Function, formats are better handled by SAS

Explicit Passthrough optimizes operations because everything is processed in database

Implicit versus Explicit Pass-Through

Implicit Pass through

```
proc SQL;
  select *
    from db.countries
      INNER JOIN
        db.cust5000
        ON country=alpha2
      where countryname like '%a'
  ;
quit;
```

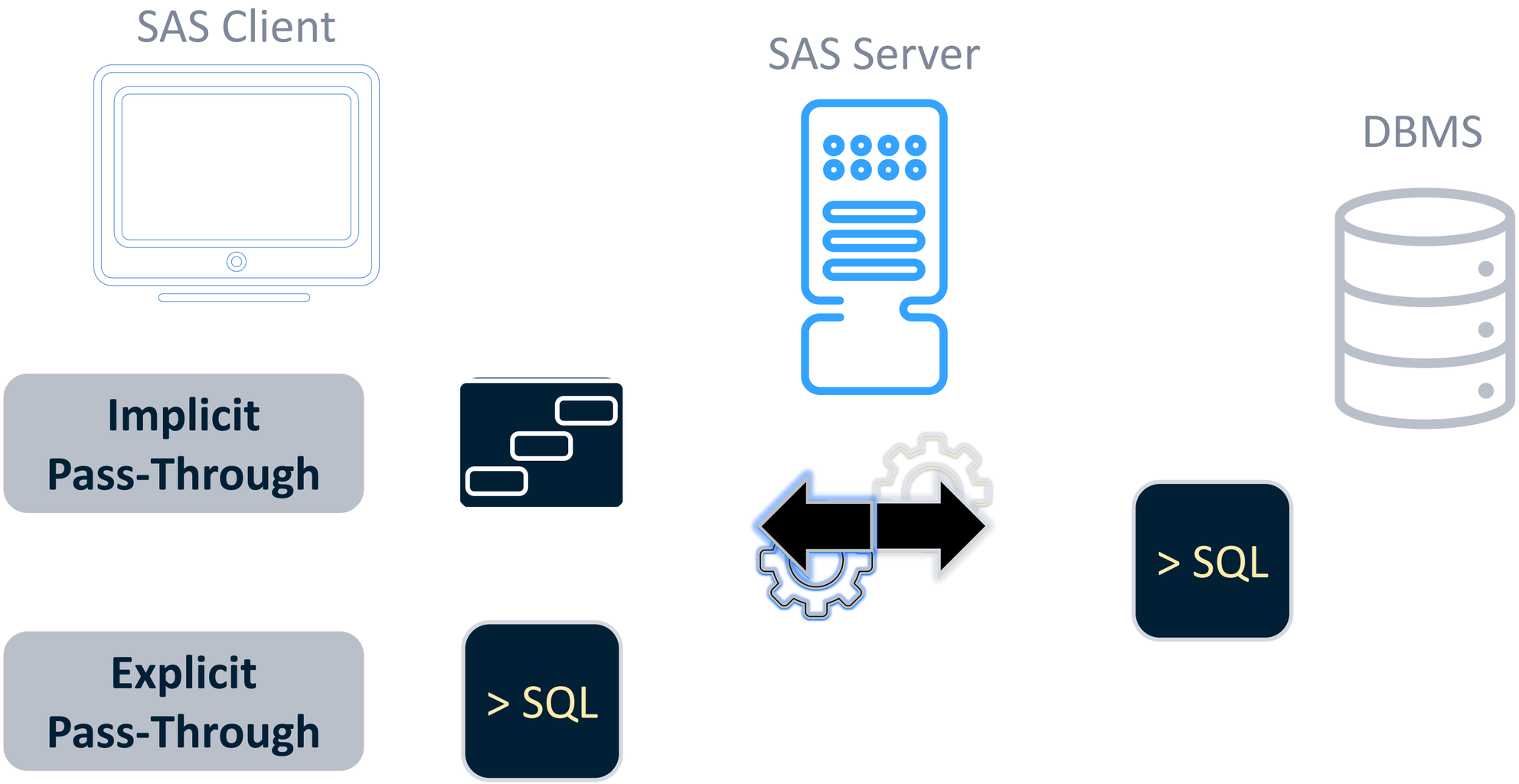
Implicit Pass Through

Explicit Pass through

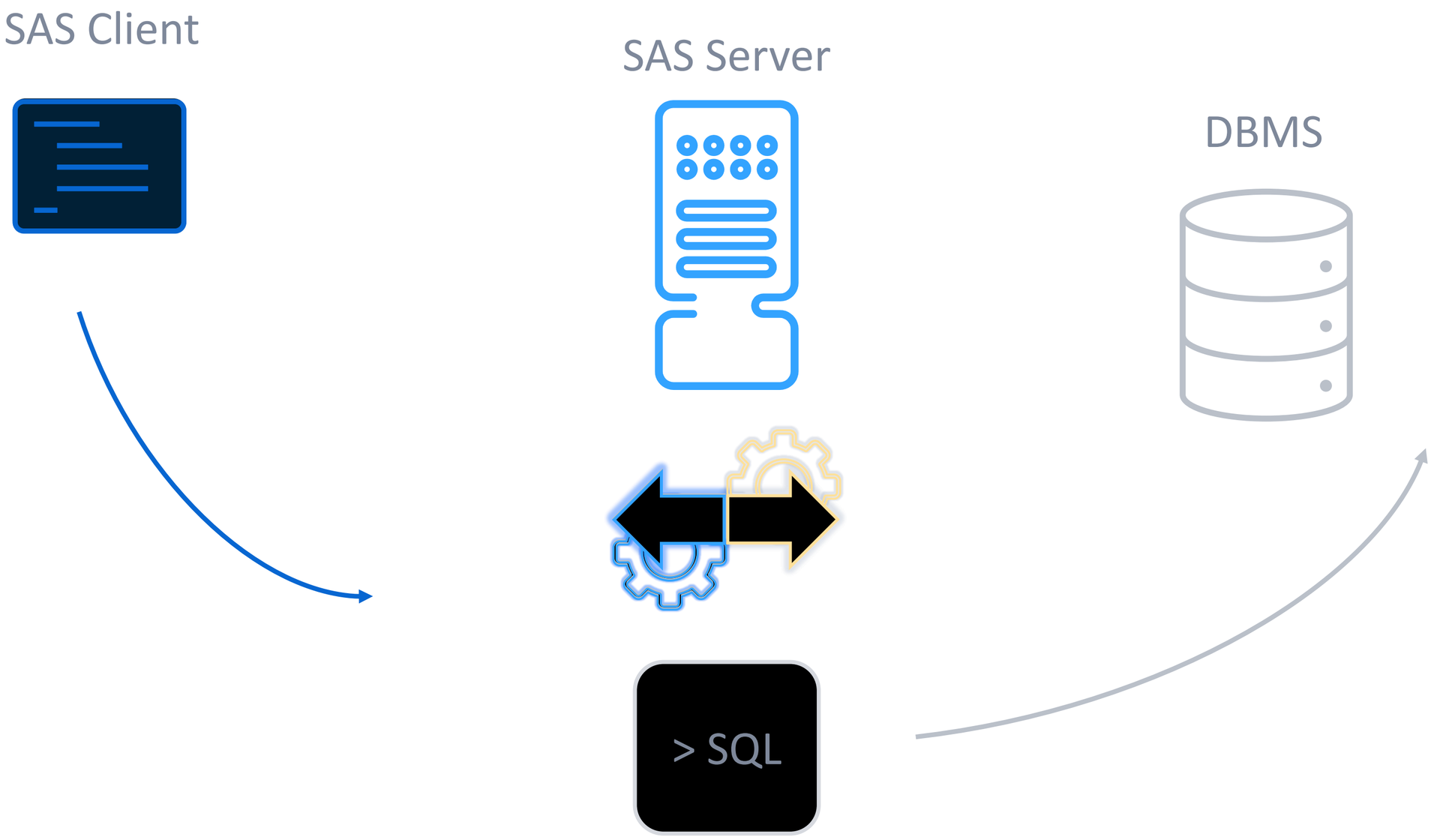
```
proc sql;
  connect to Oracle as db
(path="//server.demo.sas.com:1521/ORCL"
user=student password="Metadata0"); connect
using db;
  select *
    from connection to db
      (select *
        from etdp.countries o
          inner join
            etdp.cust5000  c
            on country=alpha2
          where countryname like '%a'
      )
  ;
disconnect from db;
quit;
```

Explicit Pass Through

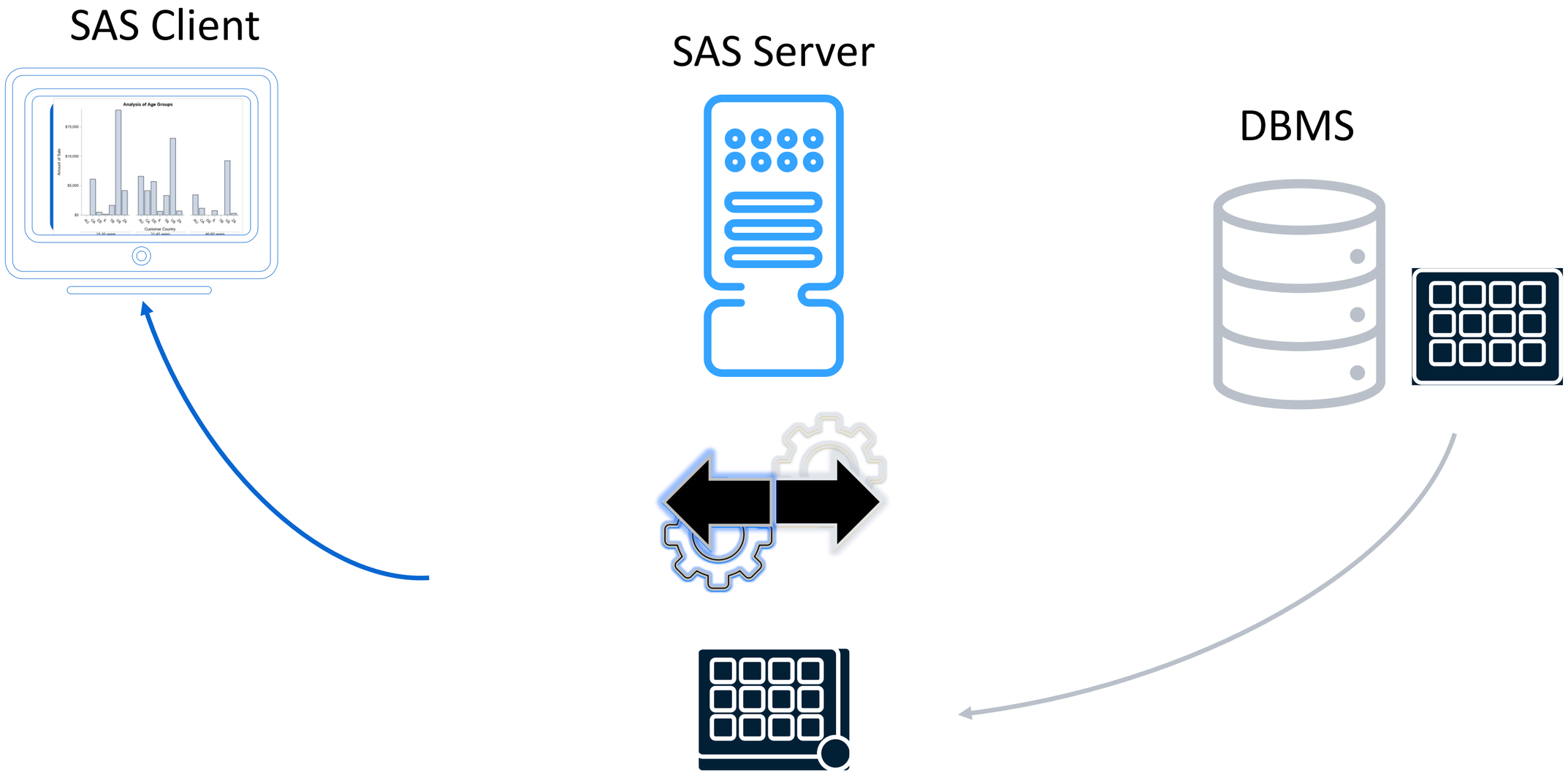
Introduction to SAS/ACCESS



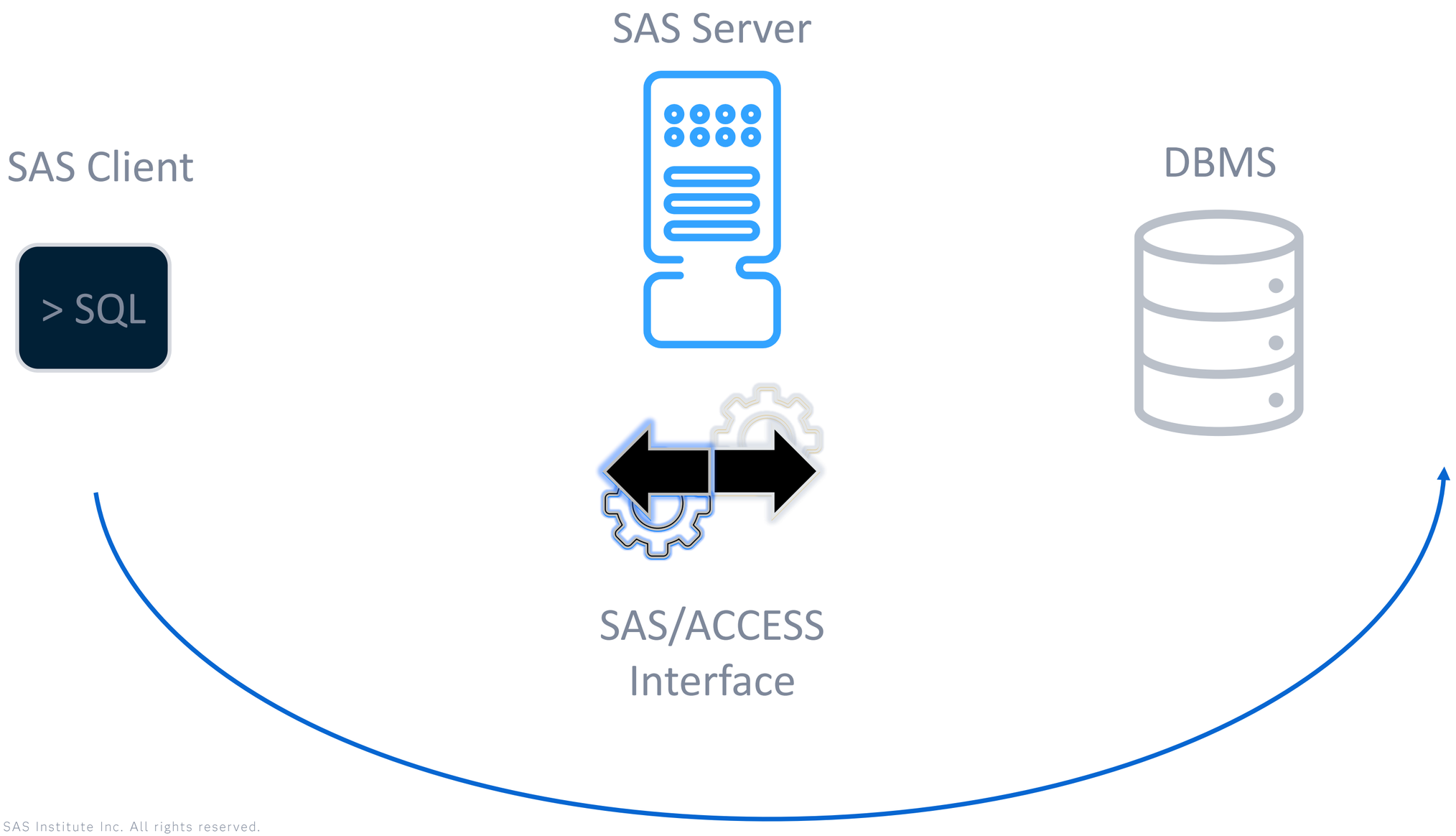
Implicit Pass-Through



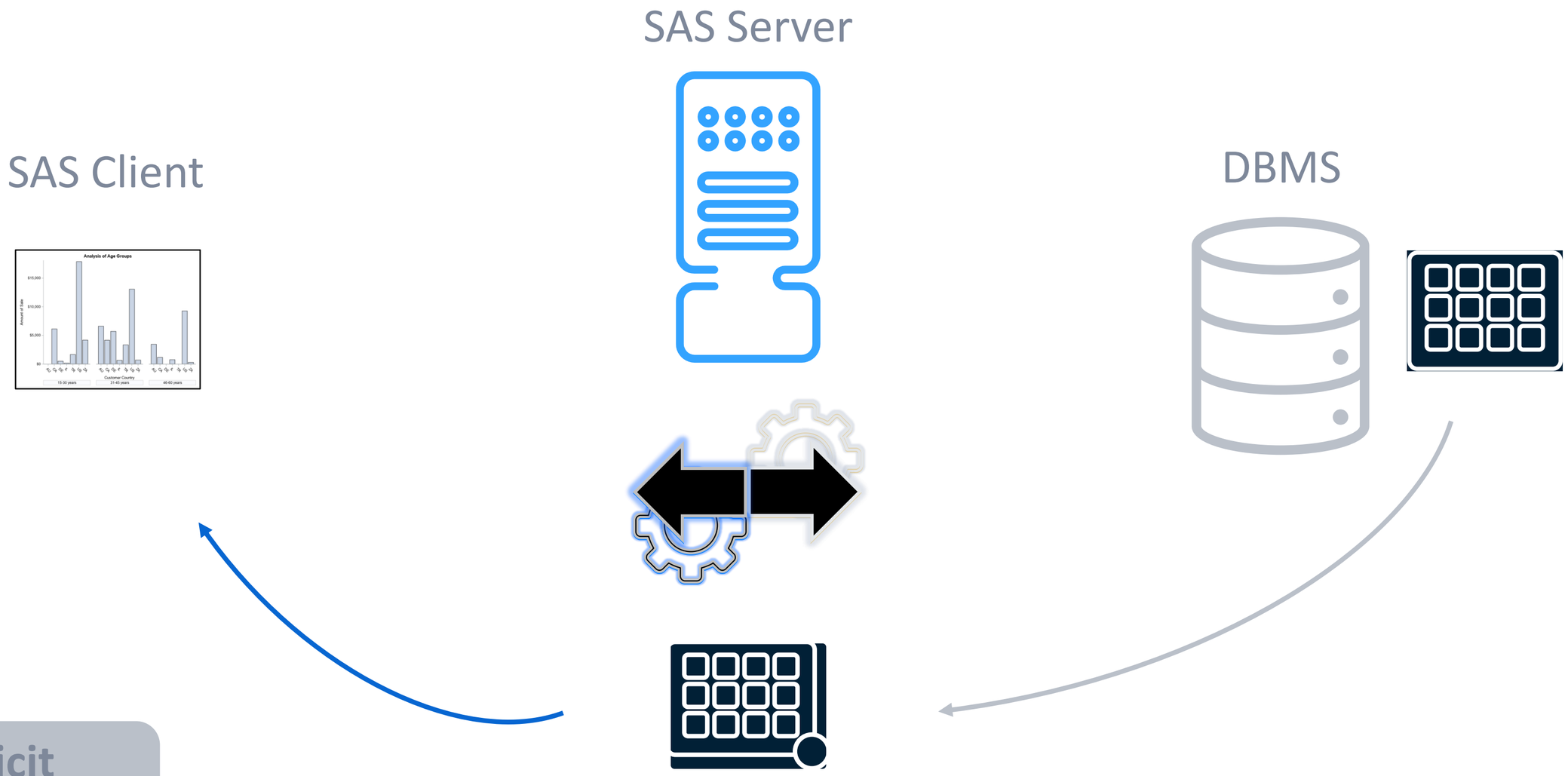
Implicit Pass Through



Explicit Pass-Through

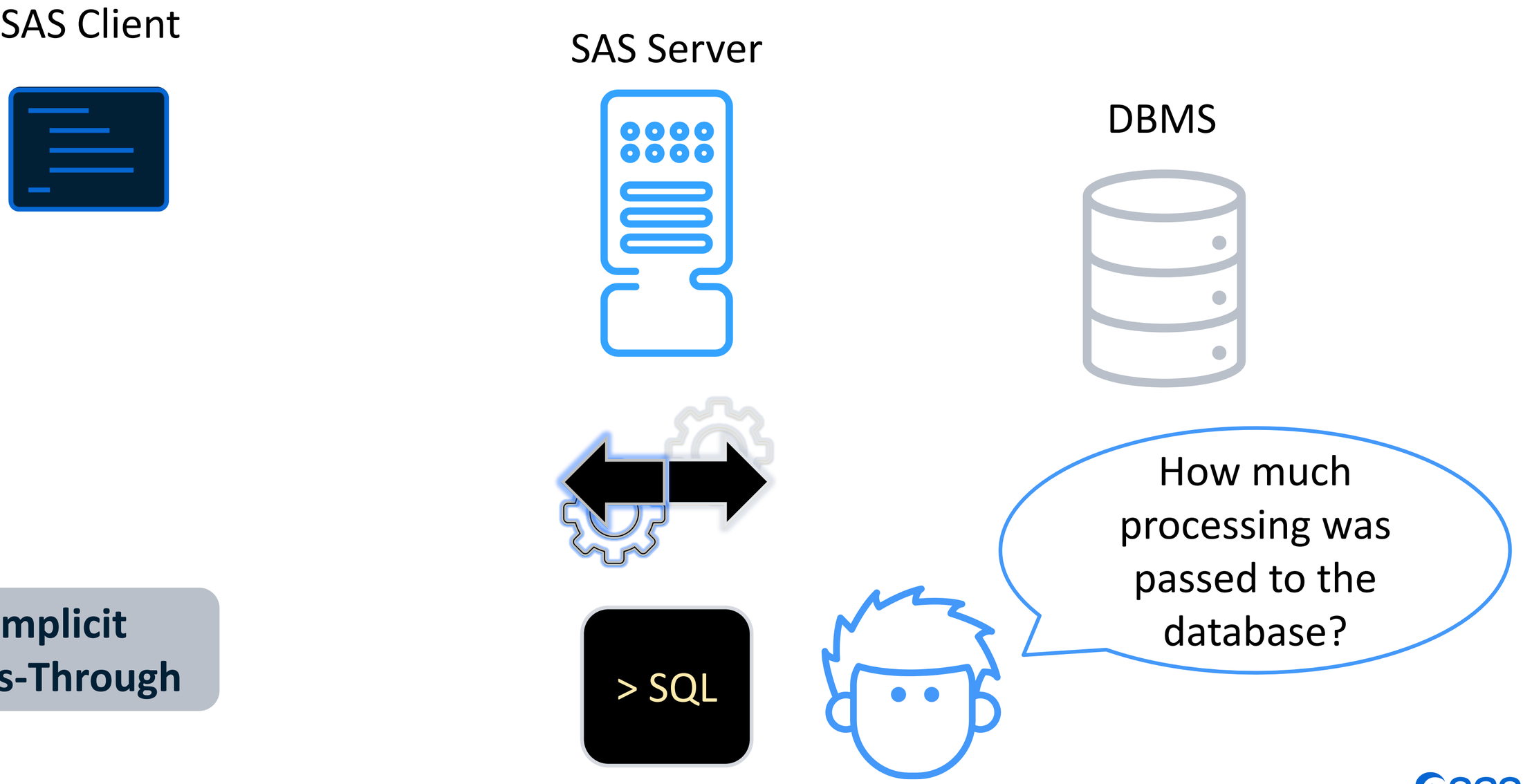


Explicit Pass Through – Return Trip



Explicit Pass-Through

Implicit Pass-Through Transparency



Implicit Pass-Through

Insight Into Operation of the SAS Access Libname Engine

DBMS_SELECT & SASTRACE SYSTEM OPTIONS

OPTIONS DEBUG=DBMS_SELECT SASTRACE=OFF | ',,,d'
SASTRACELOC=stdout | SASLOG | FILE 'path-and-file-name'
NOSTSUFFIX;

```
options sastrace=',,,d' sastracelog=saslog nostsuffix;
```

.

Query Processed on the Database – Implicit Pass Through

```
80   options debug=dbms sastrace=',,,d sastraceloc=saslog nostsuffix';
81   proc SQL;
82   create table sastable as
83   select *
84      from db.countries
85   INNER JOIN
86   db.cust5000
87   ON country=alpha2
88      where countryname like '%a'
89   ;
ORACLE: SELECT * FROM ETDP.COUNTRIES

ORACLE_510: Prepared: on connection 0
SELECT * FROM ETDP.COUNTRIES

ORACLE: SELECT * FROM ETDP.CUST5000

ORACLE_511: Prepared: on connection 0
SELECT * FROM ETDP.CUST5000

ORACLE:  select TXT_1."COUNTRYNAME", TXT_1."ALPHA2", TXT_1."ALPHA3", TXT_1."UNCODE", TXT_2."CUSTOMER_ID", TXT_2."CUSTOMER_TYPE_ID",
TXT_2."CUSTOMER_NAME", TXT_2."CUSTOMER_ADDRESS", TXT_2."CITY_NAME", TXT_2."CUSTOMER_STATE", TXT_2."POSTAL_CODE", TXT_2."COUNTRY",
TXT_2."GENDER", TXT_2."CUSTOMER_AGE", TXT_2."BIRTH_DATE", TXT_2."FIRST_PURCHASE_DATE", TXT_2."LAST_PURCHASE_DATE" from
ETDP.COUNTRIES TXT_1 inner join ETDP.CUST5000 TXT_2 on TXT_2."COUNTRY" = TXT_1."ALPHA2" where TXT_1."COUNTRYNAME" like '%a'

ORACLE_512: Prepared: on connection 0
 select TXT_1."COUNTRYNAME", TXT_1."ALPHA2", TXT_1."ALPHA3", TXT_1."UNCODE", TXT_2."CUSTOMER_ID", TXT_2."CUSTOMER_TYPE_ID",
TXT_2."CUSTOMER_NAME", TXT_2."CUSTOMER_ADDRESS", TXT_2."CITY_NAME", TXT_2."CUSTOMER_STATE", TXT_2."POSTAL_CODE", TXT_2."COUNTRY",
TXT_2."GENDER", TXT_2."CUSTOMER_AGE", TXT_2."BIRTH_DATE", TXT_2."FIRST_PURCHASE_DATE", TXT_2."LAST_PURCHASE_DATE" from
ETDP.COUNTRIES TXT_1 inner join ETDP.CUST5000 TXT_2 on TXT_2."COUNTRY" = TXT_1."ALPHA2" where TXT_1."COUNTRYNAME" like '%a'

DEBUG: SQL Implicit Passthru stmt has been prepared successfully.

DEBUG: SQL Implicit Passthru stmt used for fetching data.
ACCESS ENGINE:  SQL statement was passed to the DBMS for fetching data.
```



```
NOTE: Table WORK.SASTABLE created, with 2364 rows and 17 columns.
```

SAS TRACE highlighted in blue

DEBUG=DBMS SELECT highlighted in green

HANDY LINKS

- Top 10 SQL tricks in SAS
- Shankar, Charu A database Professional's best friend
- Tactics for Pushing SQL to the Relational Databases
- Understanding the SQL Pass-Through Available in PROC SQL
- Ask the Expert: Shankar, Charu: Top 5 Handy PROC SQL Tips
- Ask the Expert: Shankar, Charu: How Many Ways Can You Join SAS® Tables?
- Ask the Expert: Shankar, Charu: Why Choose Between SAS® DATA Step and PROC SQL When You Can Have Both?
- Ask the Expert: Shankar, Charu: Baking With Arrays Versus Cooking With Hash: In-Memory Lookup Techniques
- Johnson, Misty(2015)Just Passing Through..Or are you:Determine when SQL Pass-Through occurs to optimize your queries
- Lafler, Kirk & Shankar, Charu(2017) One-to-one One-to-many and Many-to-many Joins Using PROC SQL, Proceedings of the 2017 Western Users of SAS Software(WUSS) Conference

Thank you