Efficient Database Programming with ABAP

THE BEST-RUN BUSINESSES RUN SAP

Learning Objectives

As a result of this workshop, you will be able to:

- Explain the communication between database and application server
- Analyze bottlenecks in database programming
- Understand how table buffers and indices work
- Use Open SQL for efficient database access

Agenda

WAS and Database Architecture

ABAP Open SQL Overview

How to Identify Expensive SQL

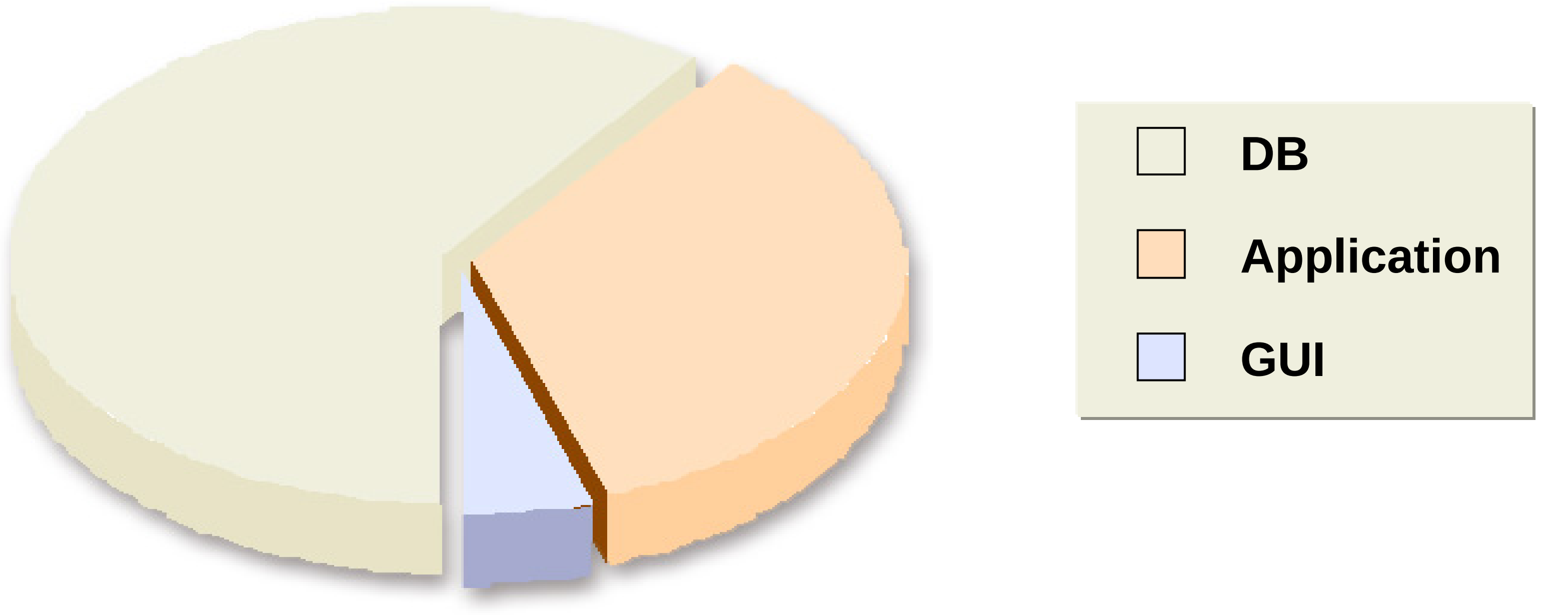
Rules for Better SQL Programming

- Theory and
- Hands-On

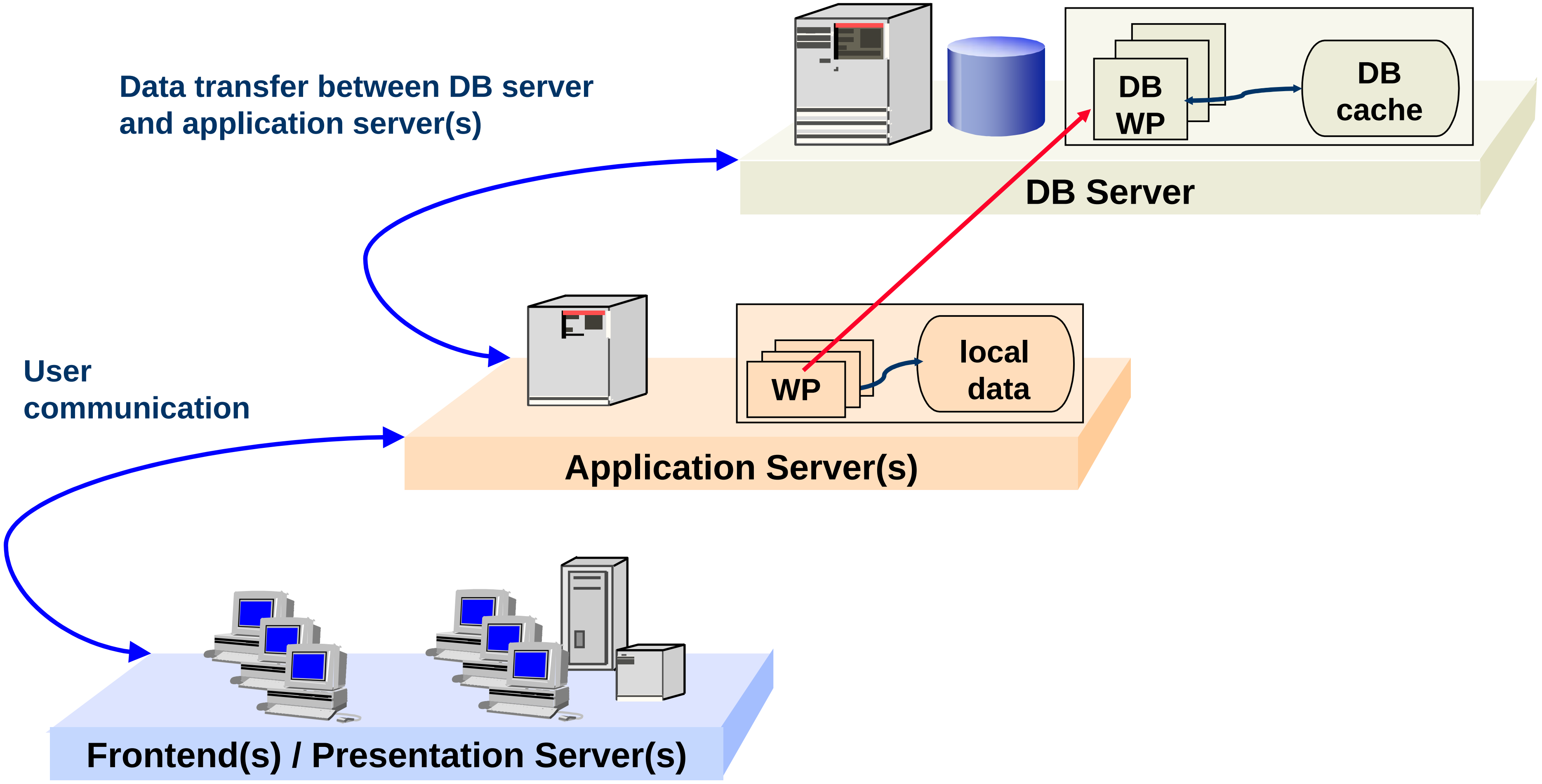
Summary

Performance of Business Transactions

General rule:
The performance of a business transaction is primarily determined by its DB accesses.



WAS and Database Architecture


Data transfer between DB server and application server(s)
User communication
DB WP
DB cache
DB Server
WP
local data
Application Server(s)
Frontend(s) / Presentation Server(s)

WAS and Database Architecture

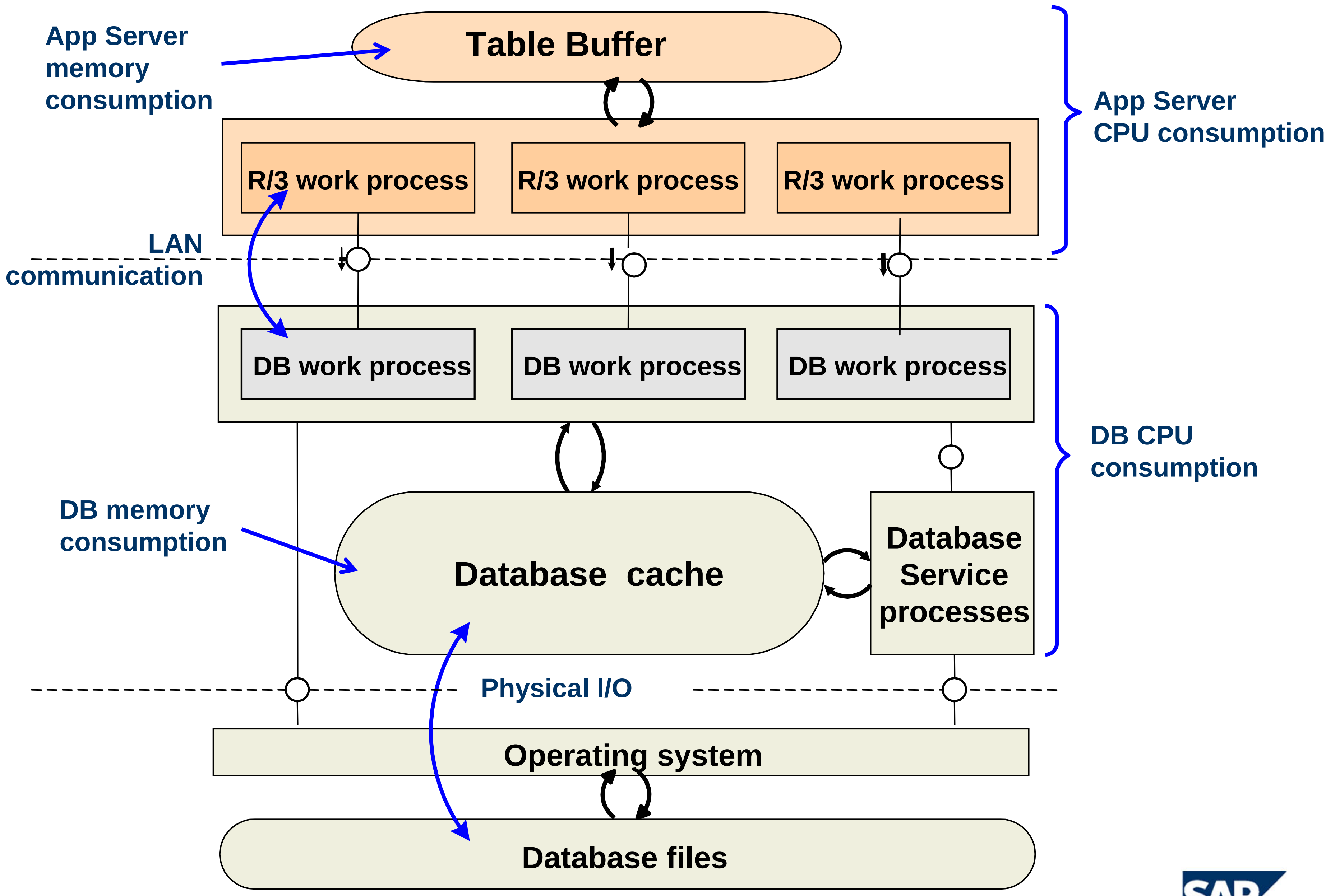


Table Buffering: Types

Single-record buffering (all key fields)

key1	key2	key3	data
001	A	2	
001	A	4	
001	B	1	
001	B	3	
001	B	5	
002	A	1	
002	A	3	
002	A	6	
002	A	8	
002	B	1	
002	**B**	**2**	
002	B	3	
002	C	0	
002	C	1	
002	D	5	
003	A	2	
003	A	3	
003	A	6	
003	B	2	
003	B	4	
003	C	5	
003	D	2	
003	D	6	
003	D	8	

Generic buffering (two key fields)

key1	key2	key3	data
001	A	2	
001	A	4	
001	B	1	
001	B	3	
001	B	5	
002	A	1	
002	A	3	
002	A	6	
002	A	8	
002	**B**	1	
002	**B**	2	
002	**B**	3	
002	C	0	
002	C	1	
002	D	5	
003	A	2	
003	A	3	
003	A	6	
003	B	2	
003	B	4	
003	C	5	
003	D	2	
003	D	6	
003	D	8	

Generic buffering (one key field)

key1	key2	key3	data
001	A	2	
001	A	4	
001	B	1	
001	B	3	
001	B	5	
002	A	1	
002	A	3	
002	A	6	
002	A	8	
002	B	1	
002	B	2	
002	B	3	
002	C	0	
002	C	1	
002	D	5	
003	A	2	
003	A	3	
003	A	6	
003	B	2	
003	B	4	
003	C	5	
003	D	2	
003	D	6	
003	D	8	

Full buffering (no key fields)

key1	key2	key3	data
001	A	2	
001	A	4	
001	B	1	
001	B	3	
001	B	5	
002	A	1	
002	A	3	
002	A	6	
002	A	8	
002	B	1	
002	B	2	
002	B	3	
002	C	0	
002	C	1	
002	D	5	
003	A	2	
003	A	3	
003	A	6	
003	B	2	
003	B	4	
003	C	5	
003	D	2	
003	D	6	
003	D	8	

Table Buffering: Invalidation Policy

Local Buffer

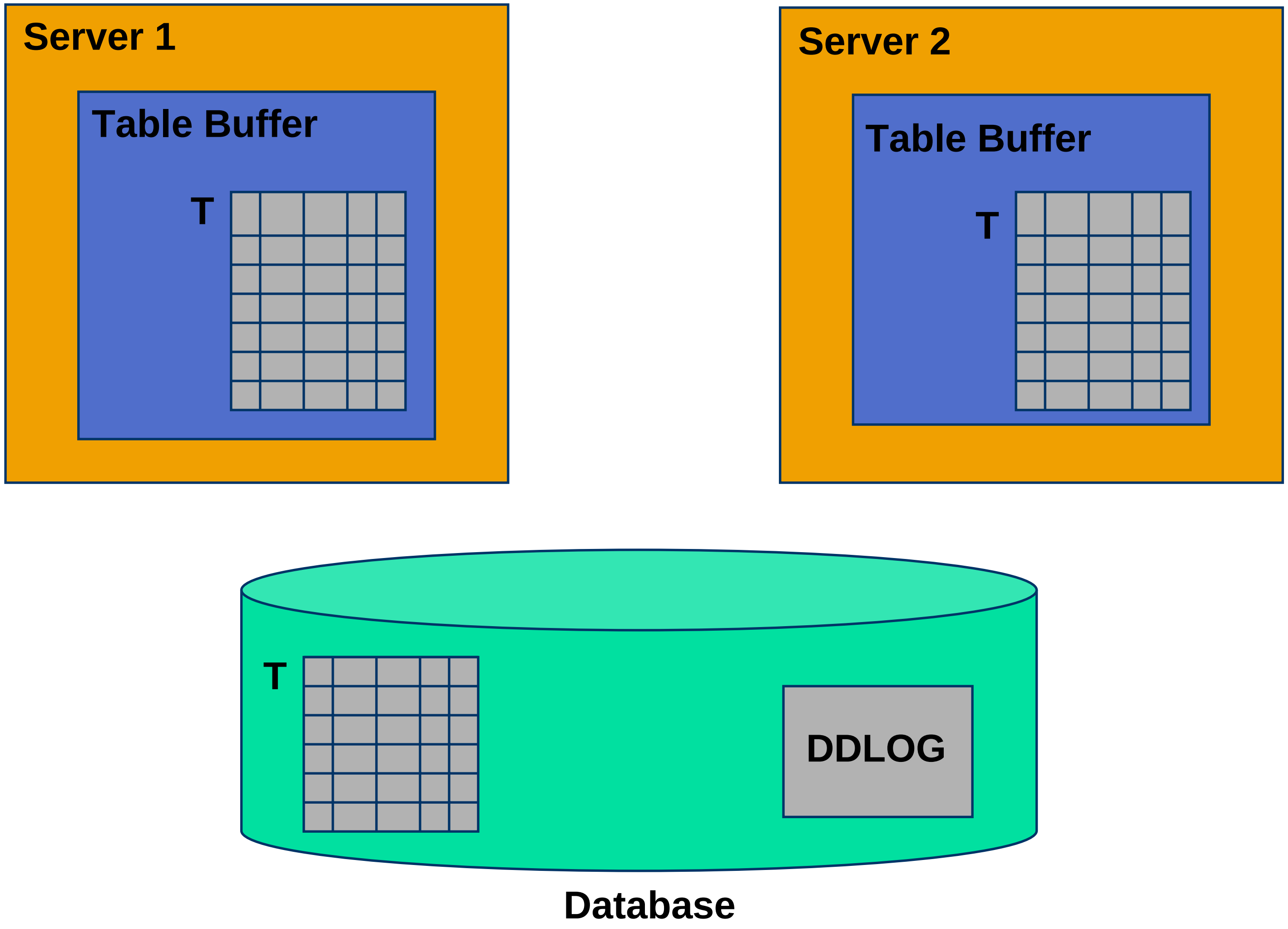
- **Full Buffering:**
  In case of a Workarea-Update, the corresponding row is updated in the buffer.
  Otherwise, the entire buffer of table t is invalidated.
- **Generic Buffering:**
  In case of a Workarea-Update, the corresponding row is updated in the buffer. If there is a change in only the generic area, only this area is invalidated. Otherwise, the entire buffer of table t is invalidated.
- **Single-Record Buffering:**
  If there is a change in only a single row, only this row is invalidated.
  Otherwise, the entire buffer of table t is invalidated.

Table Buffering: Invalidation Policy

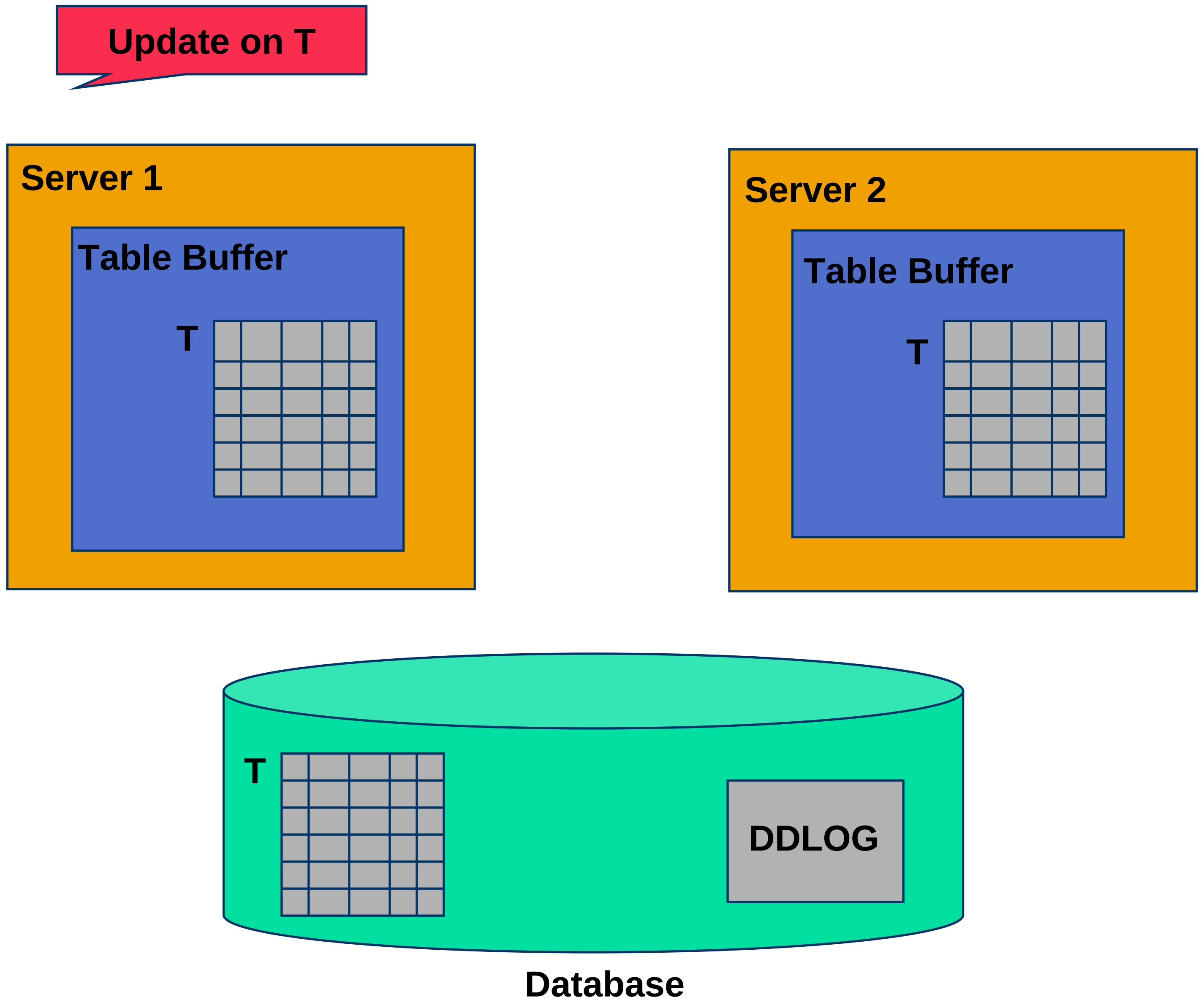
Remote Buffers

- **Full Buffering:**
  Any change invalidates the entire buffer of table *t.*
- **Generic Buffering:**
  If there is a change in only the generic area, only this area is invalidated. Otherwise, the entire buffer of table t is invalidated.
- **Single-Record Buffering:**
  If there is a change in only a single row, only this row is invalidated. Otherwise, the entire buffer of table t is invalidated.

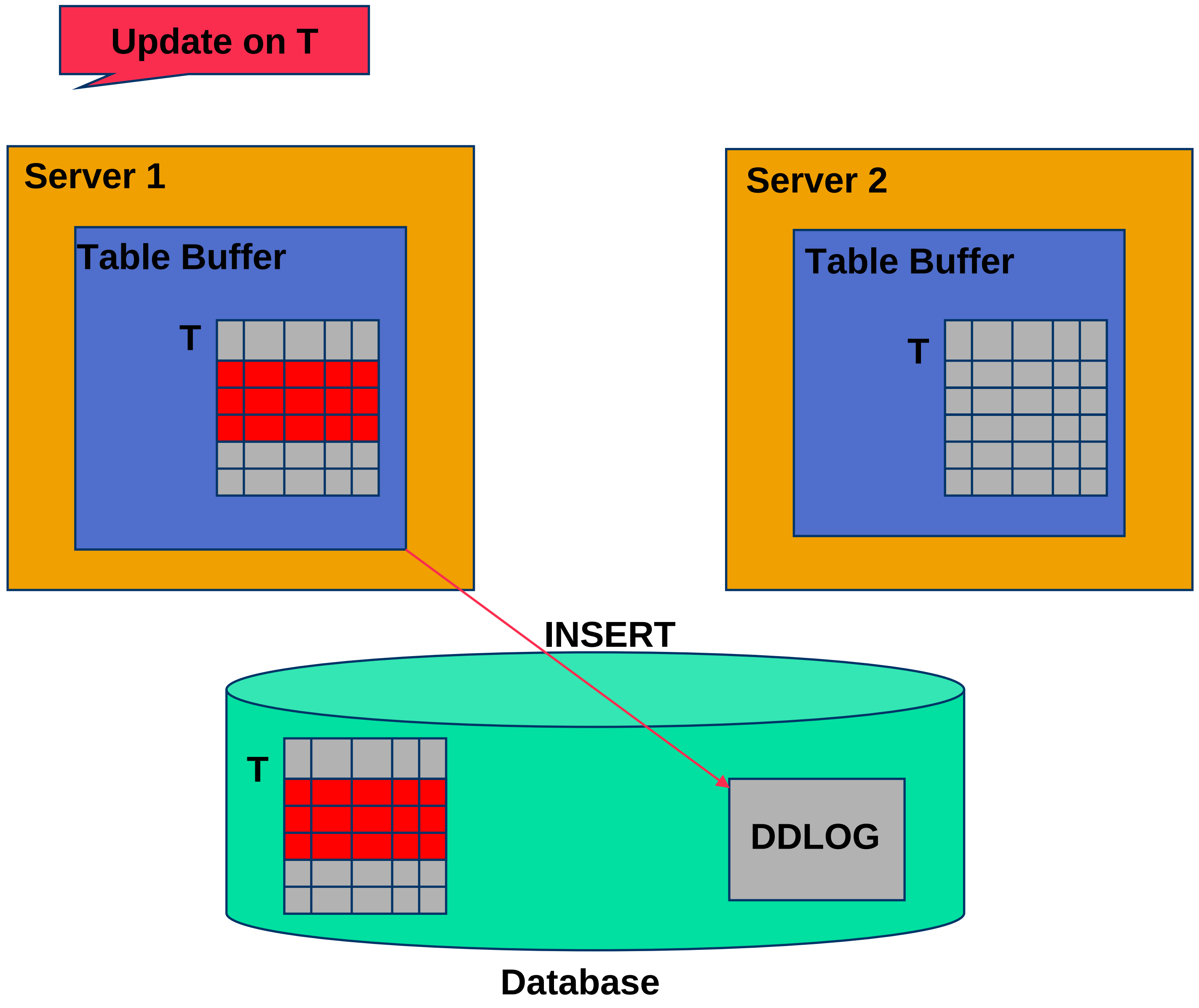
Buffer Synchronisation – Example (1)



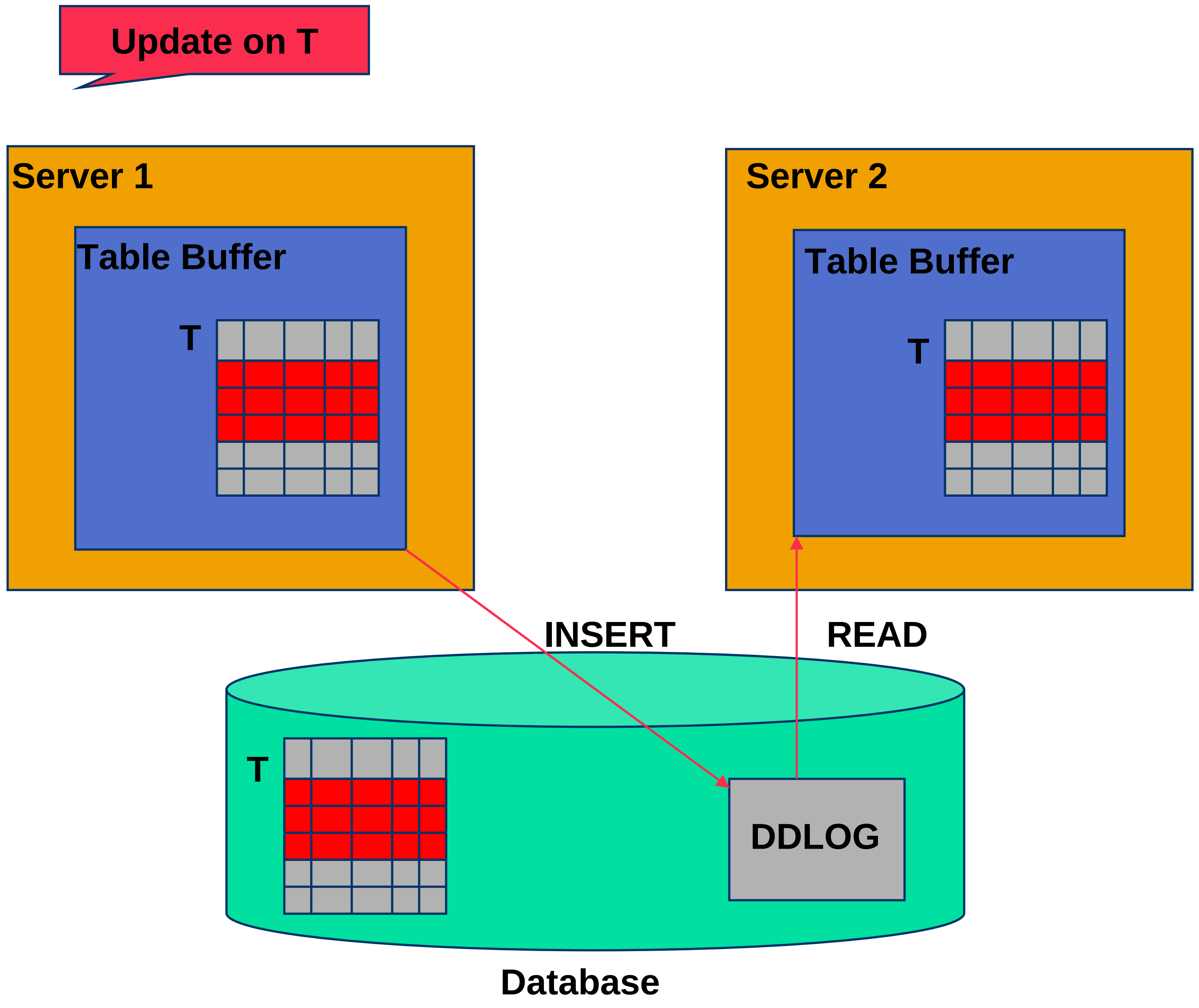
Buffer Synchronisation – Example (2)



Buffer Synchronisation – Example (3)

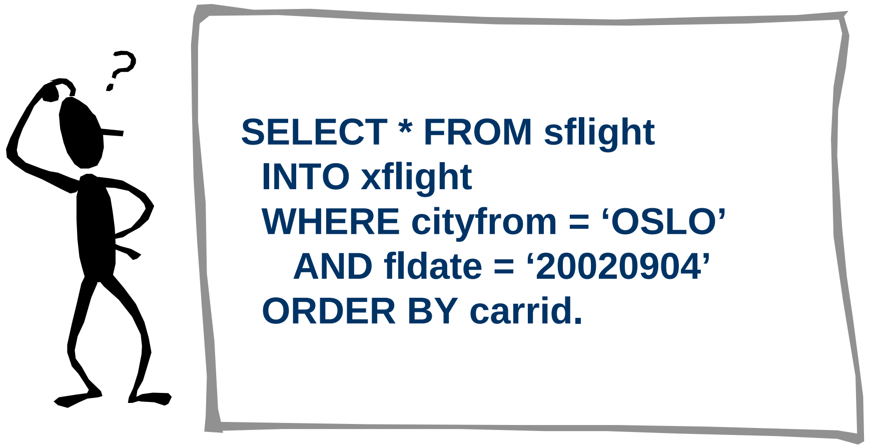


Buffer Synchronisation – Example (4)



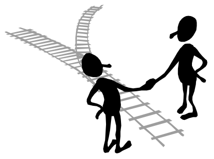
Statement Optimizer

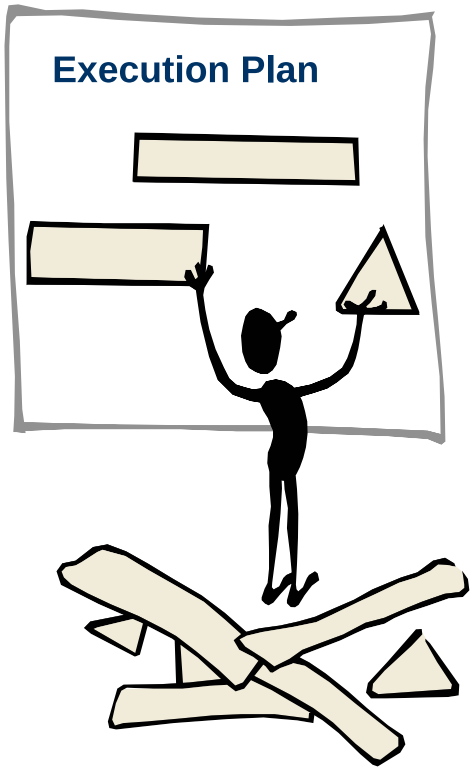
... decides how to execute the SQL statement



```
SELECT * FROM sflight
  INTO xflight
  WHERE cityfrom = ‘OSLO’
     AND fldate = ‘20020904’
  ORDER BY carrid.
```

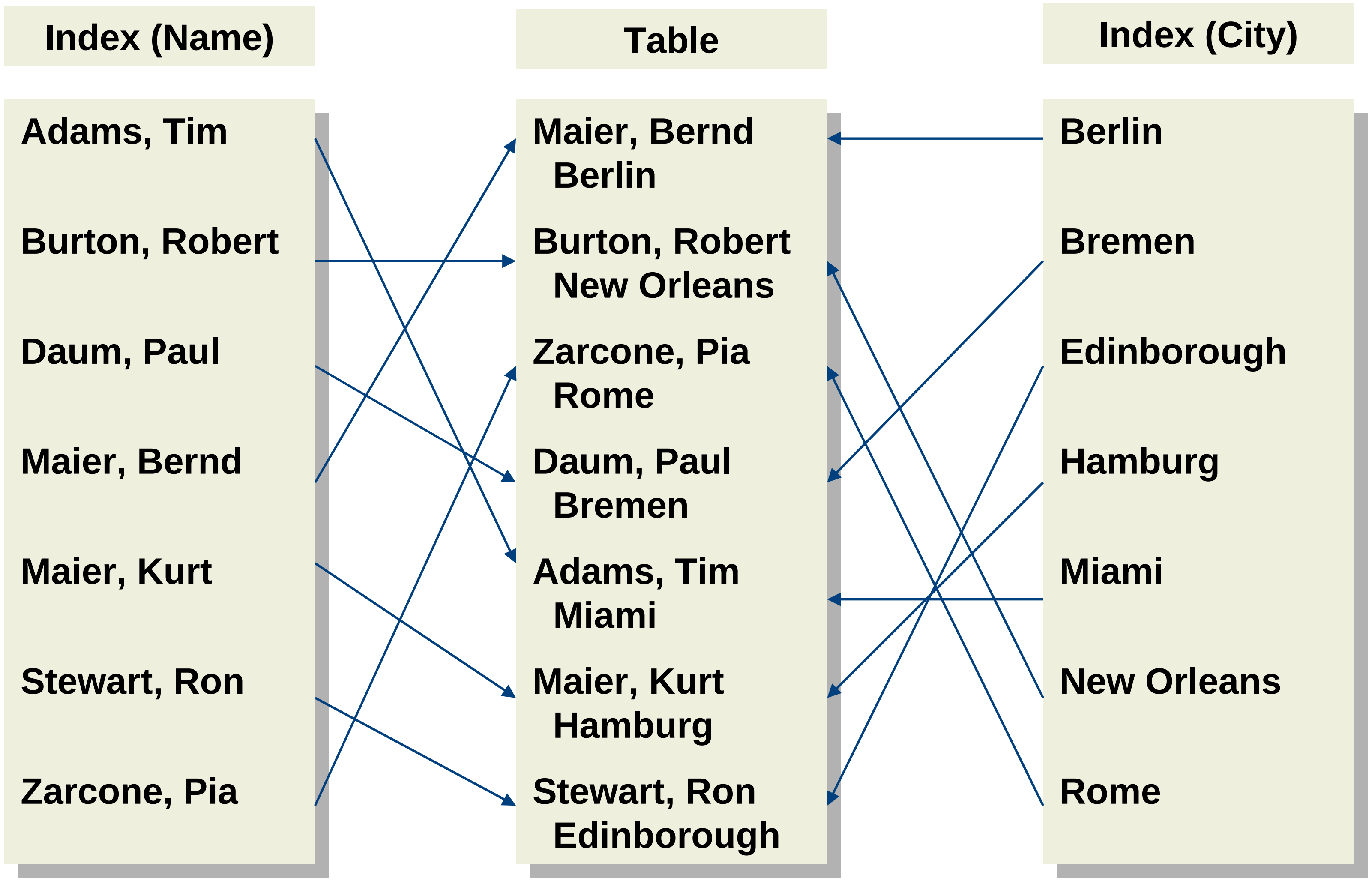
rule-based





cost-based

Tables and Indices



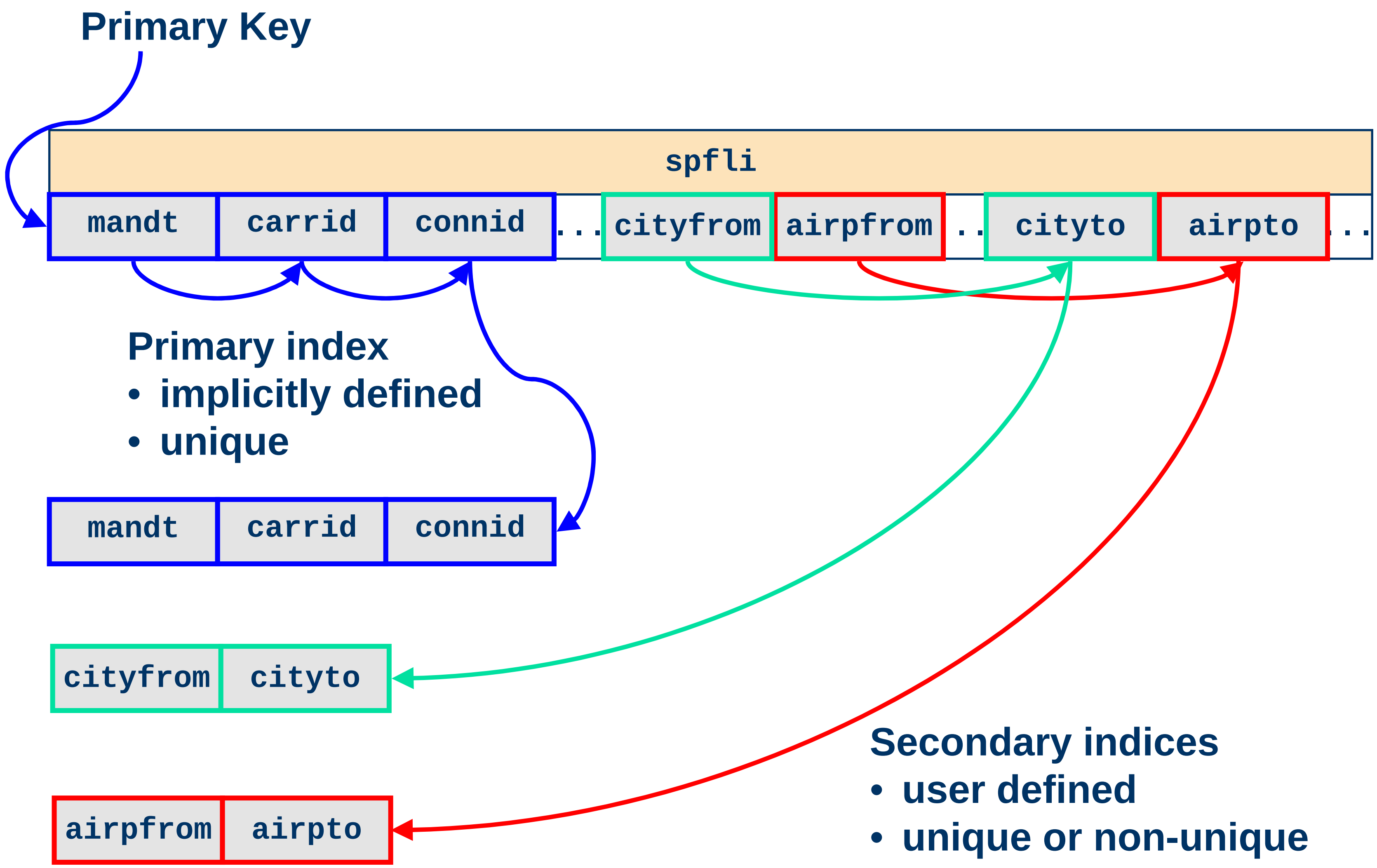
Access by Full Table Scan

```
SELECT * FROM spfli
  WHERE deptime = '150000'.
```

SPFLI

carrid	connid	airpfrom	airpto	deptime	arrtime
LH	0454	FRA	SFO	10:10	12:30
LH	0455	SFO	FRA	15:00	10:30
UA	0007	JFK	SFO	14:45	17:55
DL	1984	SFO	JFK	10:00	18:25
LH	0402	FRA	JFK	13:30	15:05
AA	0815	GKS	UNL	12:00	14:00
LH	2407	TXL	FRA	07:10	08:15
AA	0017	JFK	SFO	13:30	16:31

Tables and Indices



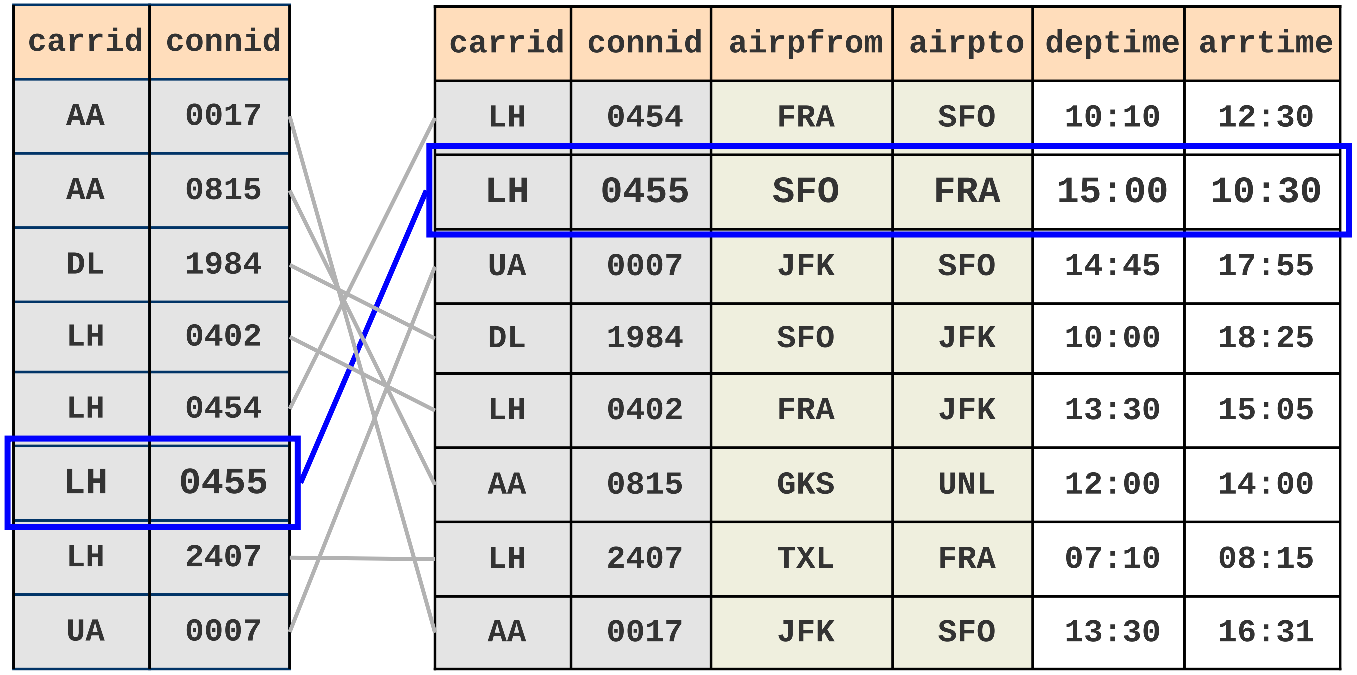
Primary index
- implicitly defined
- unique

Secondary indices
- user defined
- unique or non-unique

Access by Primary Index

```
SELECT * FROM spfli
  WHERE carrid = 'LH' AND connid = '0455'.
```

SPFLI

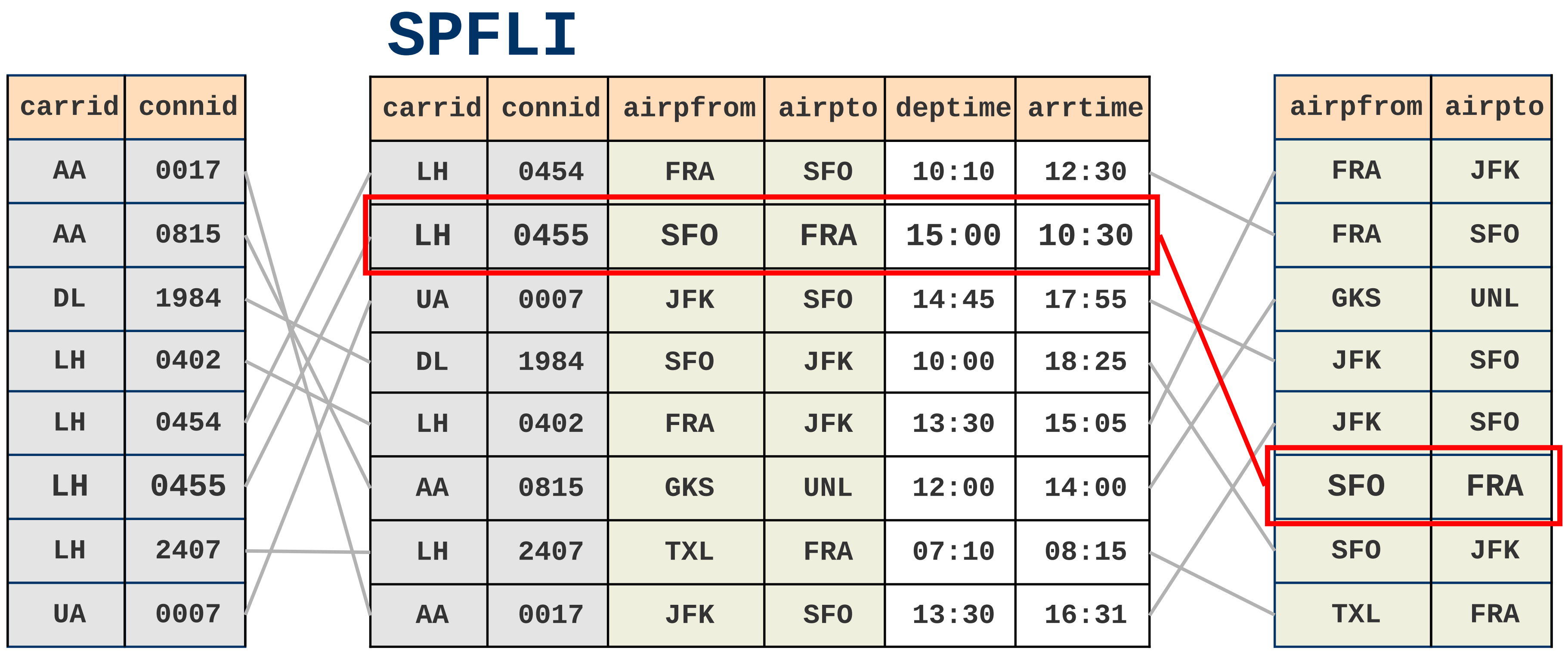


carrid	connid
AA	0017
AA	0815
DL	1984
LH	0402
LH	0454
LH	0455
LH	2407
UA	0007

carrid	connid	airpfrom	airpto	deptime	arrtime
LH	0454	FRA	SFO	10:10	12:30
LH	0455	SFO	FRA	15:00	10:30
UA	0007	JFK	SFO	14:45	17:55
DL	1984	SFO	JFK	10:00	18:25
LH	0402	FRA	JFK	13:30	15:05
AA	0815	GKS	UNL	12:00	14:00
LH	2407	TXL	FRA	07:10	08:15
AA	0017	JFK	SFO	13:30	16:31

Access by Secondary Index

```
SELECT * FROM spfli
  WHERE airpfrom = 'SFO' AND airpto = 'FRA'.
```



SPFLI

carrid	connid
AA	0017
AA	0815
DL	1984
LH	0402
LH	0454
LH	0455
LH	2407
UA	0007

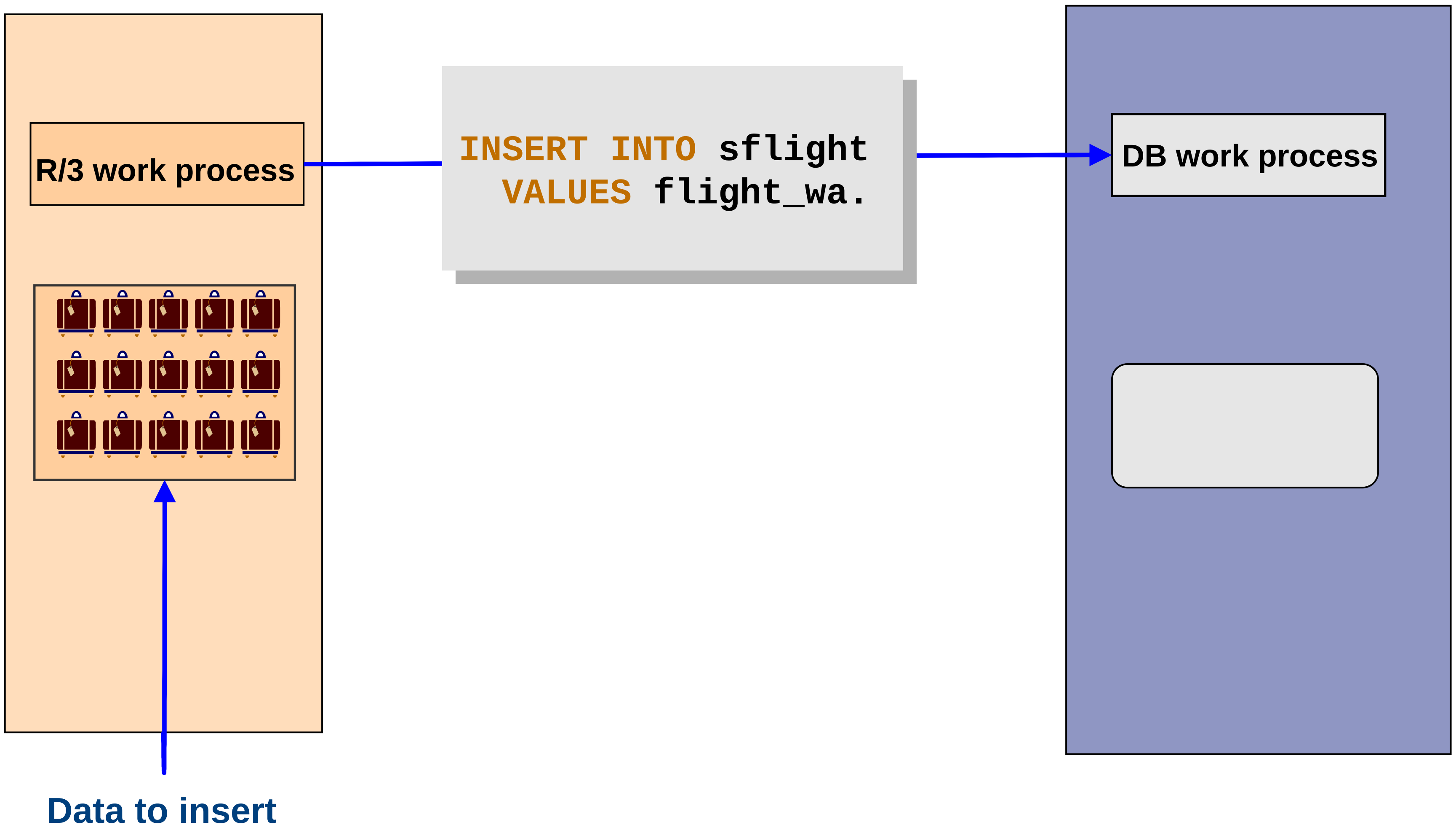
carrid	connid	airpfrom	airpto	deptime	arrtime
LH	0454	FRA	SFO	10:10	12:30
LH	0455	SFO	FRA	15:00	10:30
UA	0007	JFK	SFO	14:45	17:55
DL	1984	SFO	JFK	10:00	18:25
LH	0402	FRA	JFK	13:30	15:05
AA	0815	GKS	UNL	12:00	14:00
LH	2407	TXL	FRA	07:10	08:15
AA	0017	JFK	SFO	13:30	16:31

airpfrom	airpto
FRA	JFK
FRA	SFO
GKS	UNL
JFK	SFO
JFK	SFO
SFO	FRA
SFO	JFK
TXL	FRA

Indices

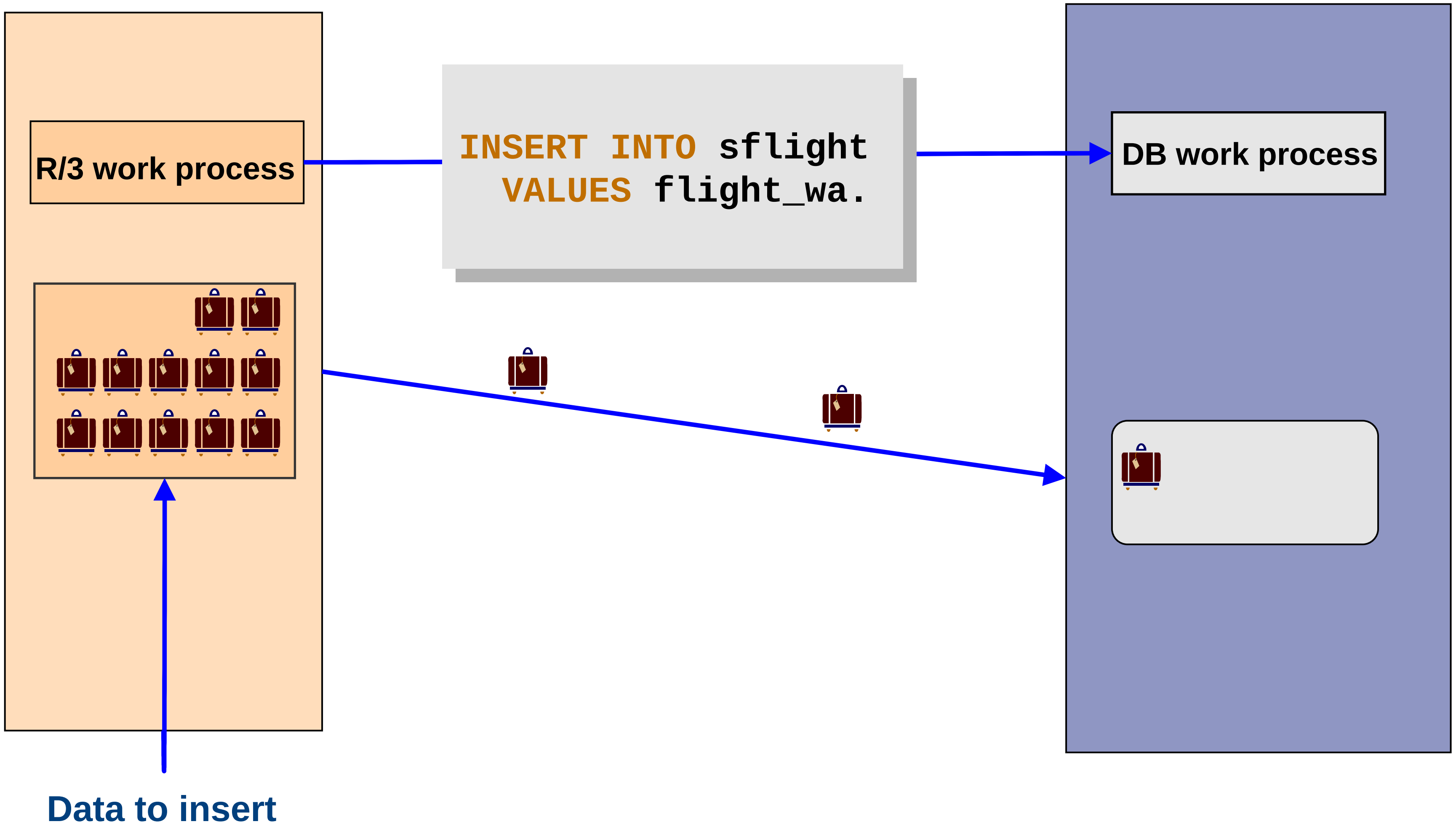
- Indices can vastly improve performance when searching for data.
- Indices will slightly slow down updates.
- A bad index is worse than none at all because data blocks might be read again and again.
- Sometimes a Full Table Scan is faster. The optimizer should do it right.

Data Transfer: Single Row INSERT



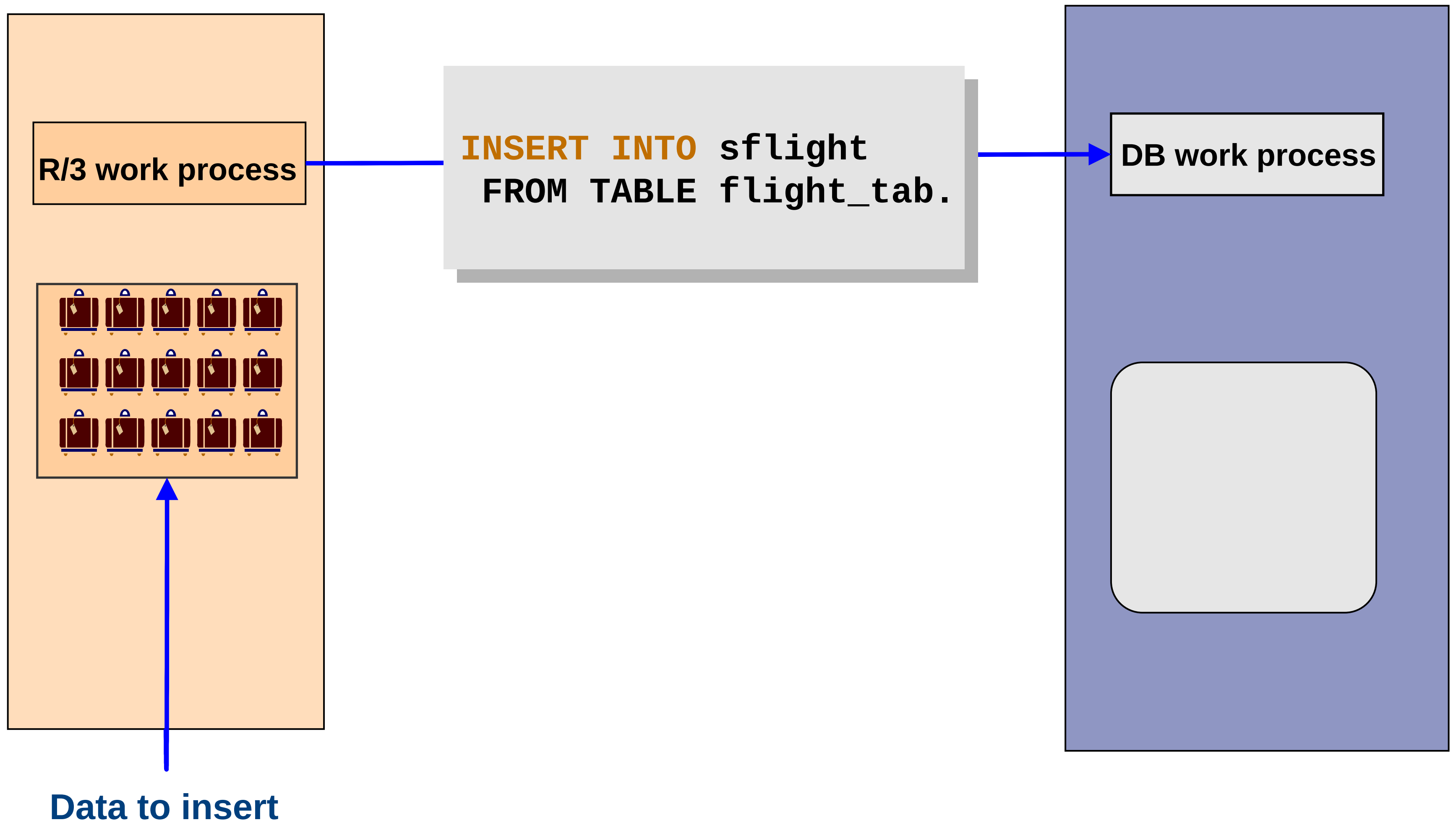
```
INSERT INTO sflight
  VALUES flight_wa.
```

Data Transfer: Single Row INSERT



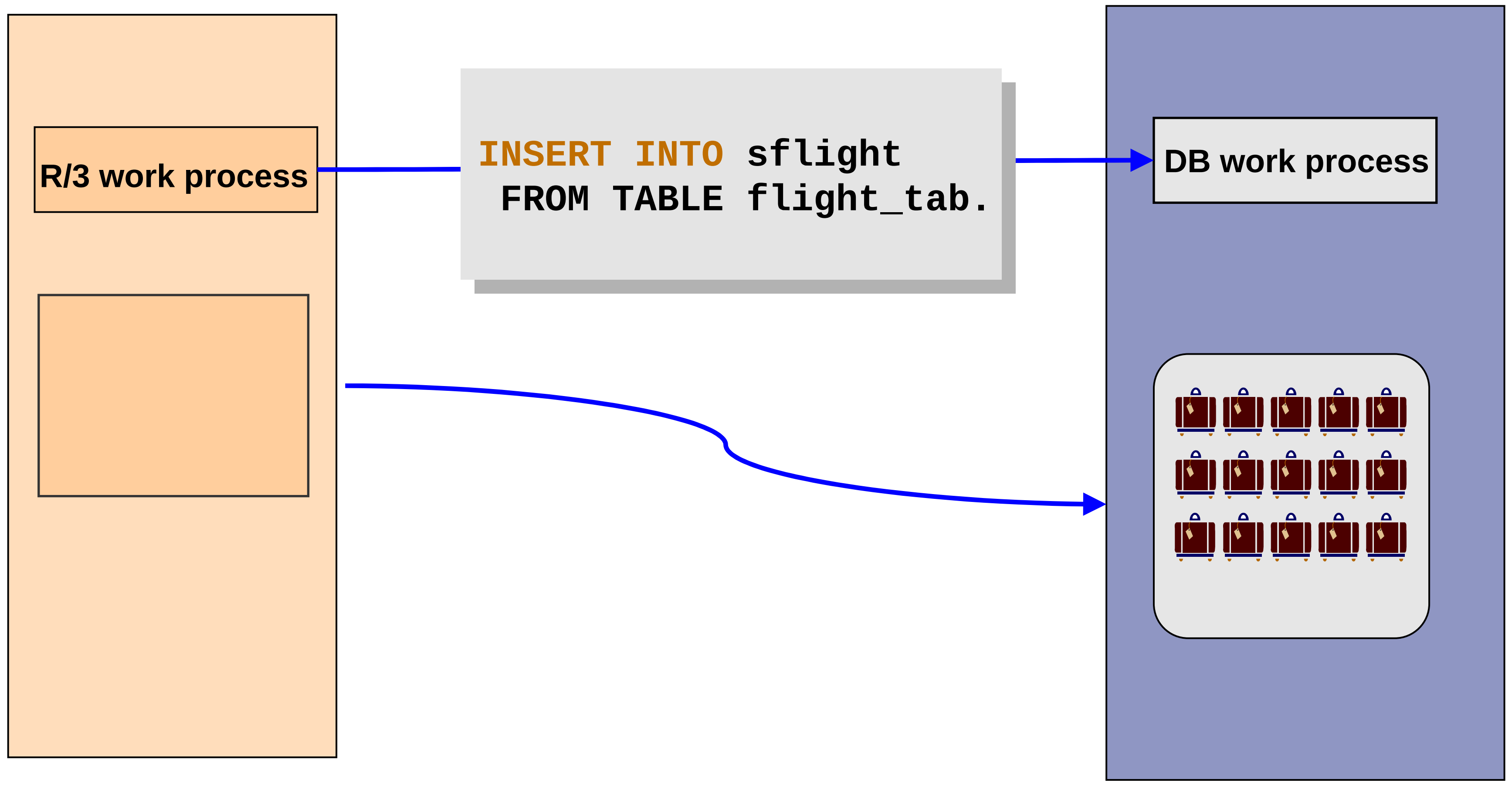
```
INSERT INTO sflight
  VALUES flight_wa.
```

Data Transfer: Array INSERT



```
INSERT INTO sflight
 FROM TABLE flight_tab.
```

Data Transfer: Array INSERT



```
INSERT INTO sflight
 FROM TABLE flight_tab.
```

THE BEST-RUN BUSINESSES RUN SAP

Agenda

WAS and Database Architecture

ABAP Open SQL Overview

How to Identify Expensive SQL

Rules for Better SQL Programming

- Theory and
- Hands-On

Summary

Open SQL Statements

SELECT

OPEN CURSOR, FETCH NEXT, CLOSE CURSOR

INSERT

UPDATE

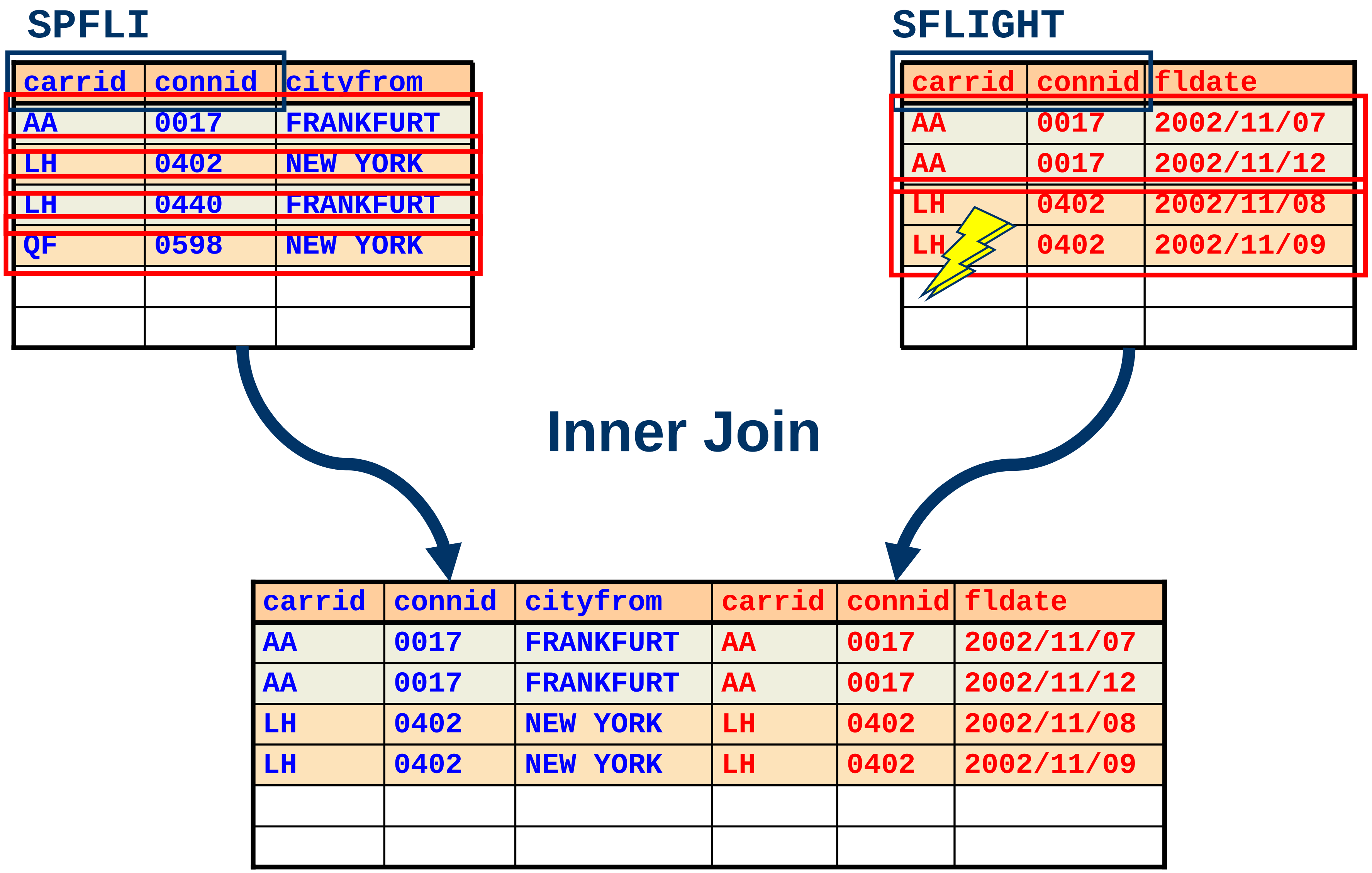
MODIFY

DELETE

COMMIT WORK

ROLLBACK WORK

Inner Join



SPFLI

carrid	connid	cityfrom
AA	0017	FRANKFURT
LH	0402	NEW YORK
LH	0440	FRANKFURT
QF	0598	NEW YORK

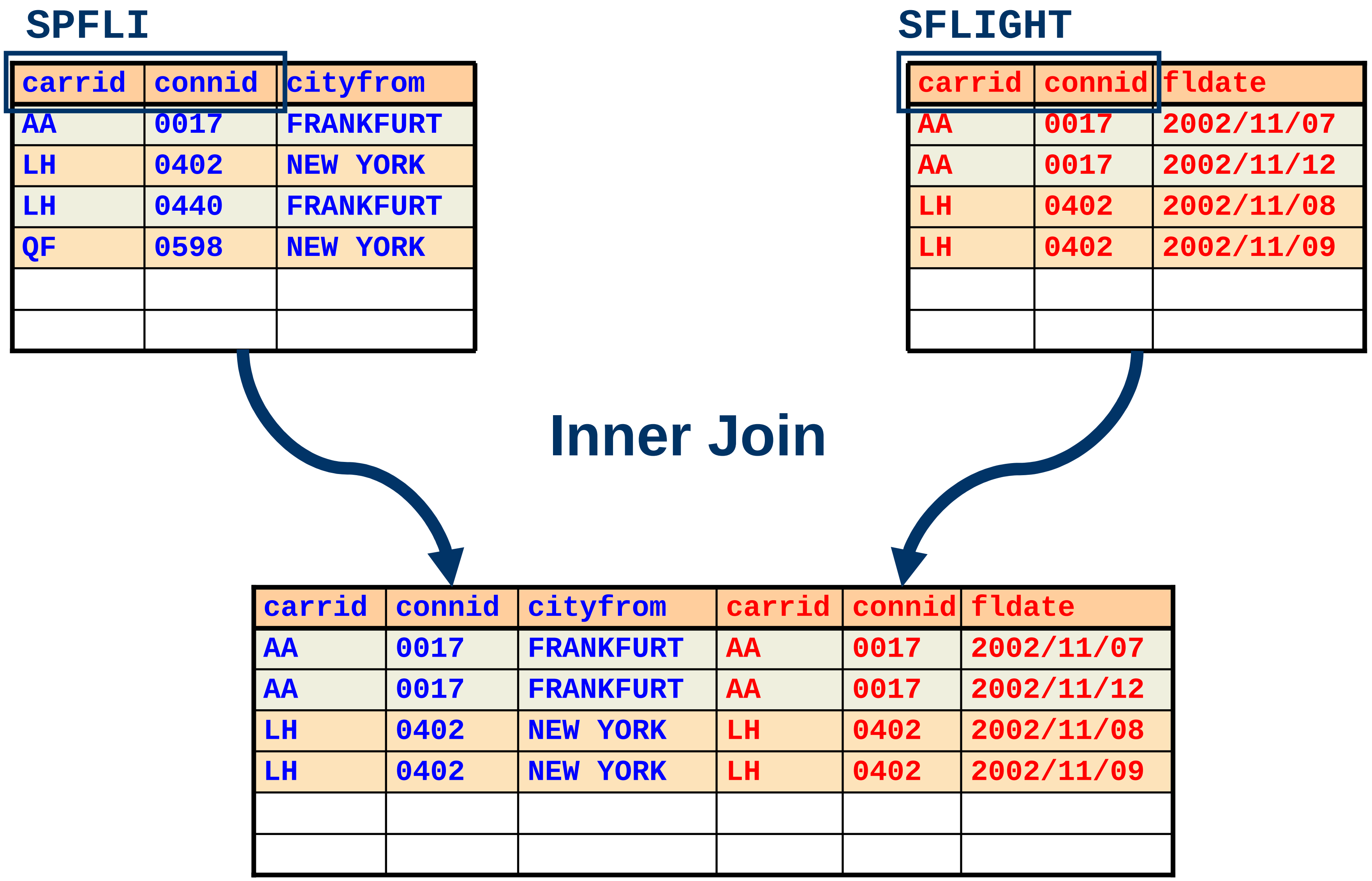
SFLIGHT

carrid	connid	fldate
AA	0017	2002/11/07
AA	0017	2002/11/12
LH	0402	2002/11/08
LH	0402	2002/11/09

Inner Join

carrid	connid	cityfrom	carrid	connid	fldate
AA	0017	FRANKFURT	AA	0017	2002/11/07
AA	0017	FRANKFURT	AA	0017	2002/11/12
LH	0402	NEW YORK	LH	0402	2002/11/08
LH	0402	NEW YORK	LH	0402	2002/11/09

Inner Join



SPFLI

carrid	connid	cityfrom
AA	0017	FRANKFURT
LH	0402	NEW YORK
LH	0440	FRANKFURT
QF	0598	NEW YORK

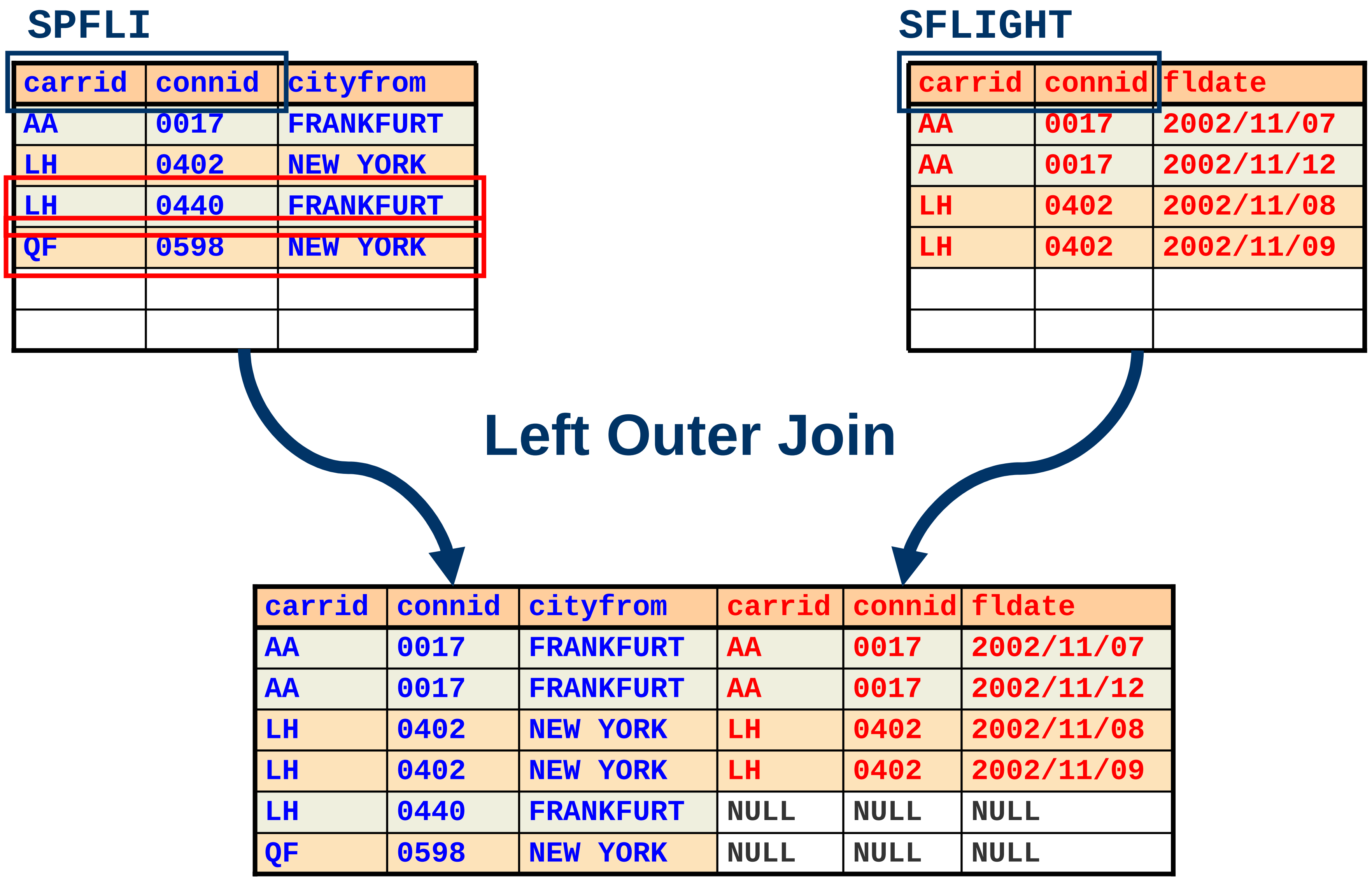
SFLIGHT

carrid	connid	fldate
AA	0017	2002/11/07
AA	0017	2002/11/12
LH	0402	2002/11/08
LH	0402	2002/11/09

Inner Join

carrid	connid	cityfrom	carrid	connid	fldate
AA	0017	FRANKFURT	AA	0017	2002/11/07
AA	0017	FRANKFURT	AA	0017	2002/11/12
LH	0402	NEW YORK	LH	0402	2002/11/08
LH	0402	NEW YORK	LH	0402	2002/11/09

Left Outer Join



SPFLI

carrid	connid	cityfrom
AA	0017	FRANKFURT
LH	0402	NEW YORK
LH	0440	FRANKFURT
QF	0598	NEW YORK

SFLIGHT

carrid	connid	fldate
AA	0017	2002/11/07
AA	0017	2002/11/12
LH	0402	2002/11/08
LH	0402	2002/11/09

Left Outer Join

carrid	connid	cityfrom	carrid	connid	fldate
AA	0017	FRANKFURT	AA	0017	2002/11/07
AA	0017	FRANKFURT	AA	0017	2002/11/12
LH	0402	NEW YORK	LH	0402	2002/11/08
LH	0402	NEW YORK	LH	0402	2002/11/09
LH	0440	FRANKFURT	NULL	NULL	NULL
QF	0598	NEW YORK	NULL	NULL	NULL

Left Outer Join

SPFLI

carrid	connid	cityfrom
AA	0017	FRANKFURT
LH	0402	NEW YORK
LH	0440	FRANKFURT
QF	0598	NEW YORK

SFLIGHT

carrid	connid	fldate
AA	0017	2002/11/07
AA	0017	2002/11/12
LH	0402	2002/11/08
LH	0402	2002/11/09

Left Outer Join

carrid	connid	cityfrom	carrid	connid	fldate
AA	0017	FRANKFURT	AA	0017	2002/11/07
AA	0017	FRANKFURT	AA	0017	2002/11/12
LH	0402	NEW YORK	LH	0402	2002/11/08
LH	0402	NEW YORK	LH	0402	2002/11/09
LH	0440	FRANKFURT	NULL	NULL	NULL
QF	0598	NEW YORK	NULL	NULL	NULL

Subqueries

Sometimes you just want to know whether some records in a secondary table exist or not. You don‘t need their actual content. This is where Subqueries come in handy:

Example: Detect an inconsistency

Are there any rows in the SFLIGHT table without a corresponding entry in table SPFLI?

```
SELECT carrid connid
 INTO (xcarrid, xconnid)
 FROM sflight AS f
 WHERE NOT EXISTS ( SELECT * FROM spfli
                      WHERE carrid = f~carrid
                        AND connid = f~connid ).
```

Agenda

WAS and Database Architecture

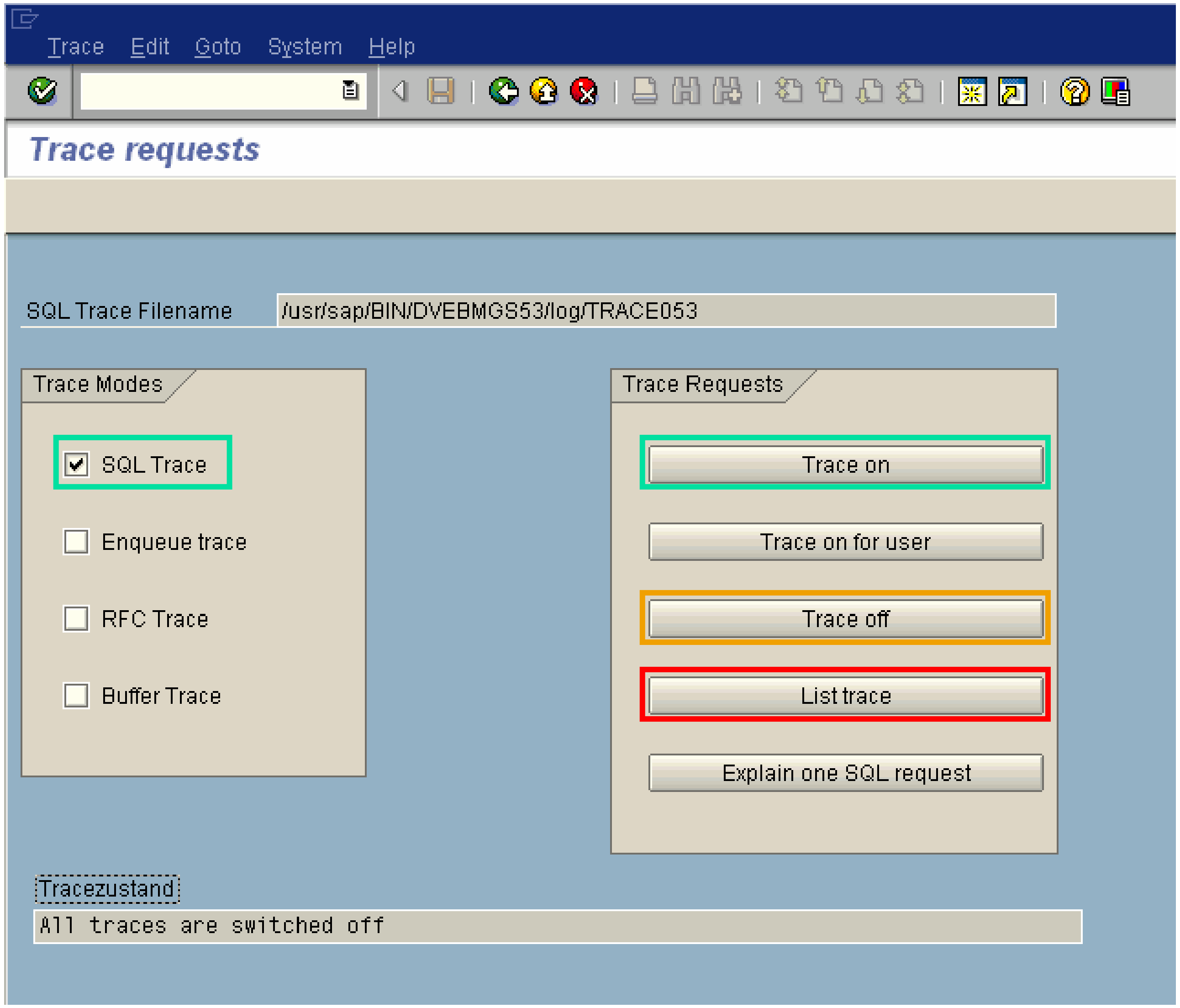
ABAP Open SQL Overview

How to Identify Expensive SQL

Rules for Better SQL Programming
- Theory and
- Hands-On

Summary

SQL Trace



Trace requests

SQL Trace Filename /usr/sap/BIN/DVEBMGS53/log/TRACE053

Trace Modes

- [✓] SQL Trace
- [ ] Enqueue trace
- [ ] RFC Trace
- [ ] Buffer Trace

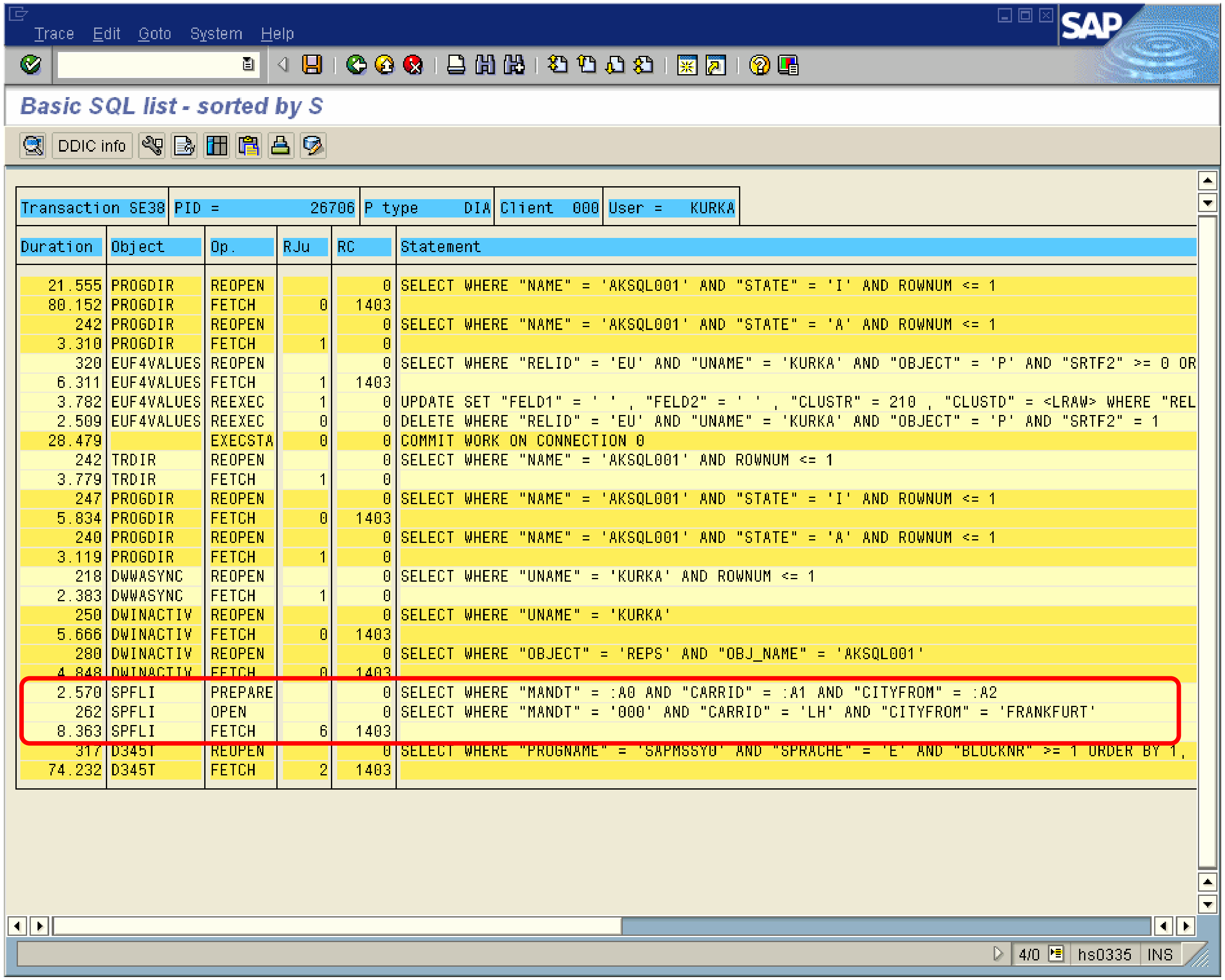
Trace Requests

Tracezustand

All traces are switched off

1. **Start ST05**
2. **Switch on the SQL trace: "*Trace on*"**
3. **Run the test program (in a different window)**
4. **Switch off the SQL trace: "*Trace off*"**
5. **List the SQL statements recorded: "*List trace*"**

SQL Trace

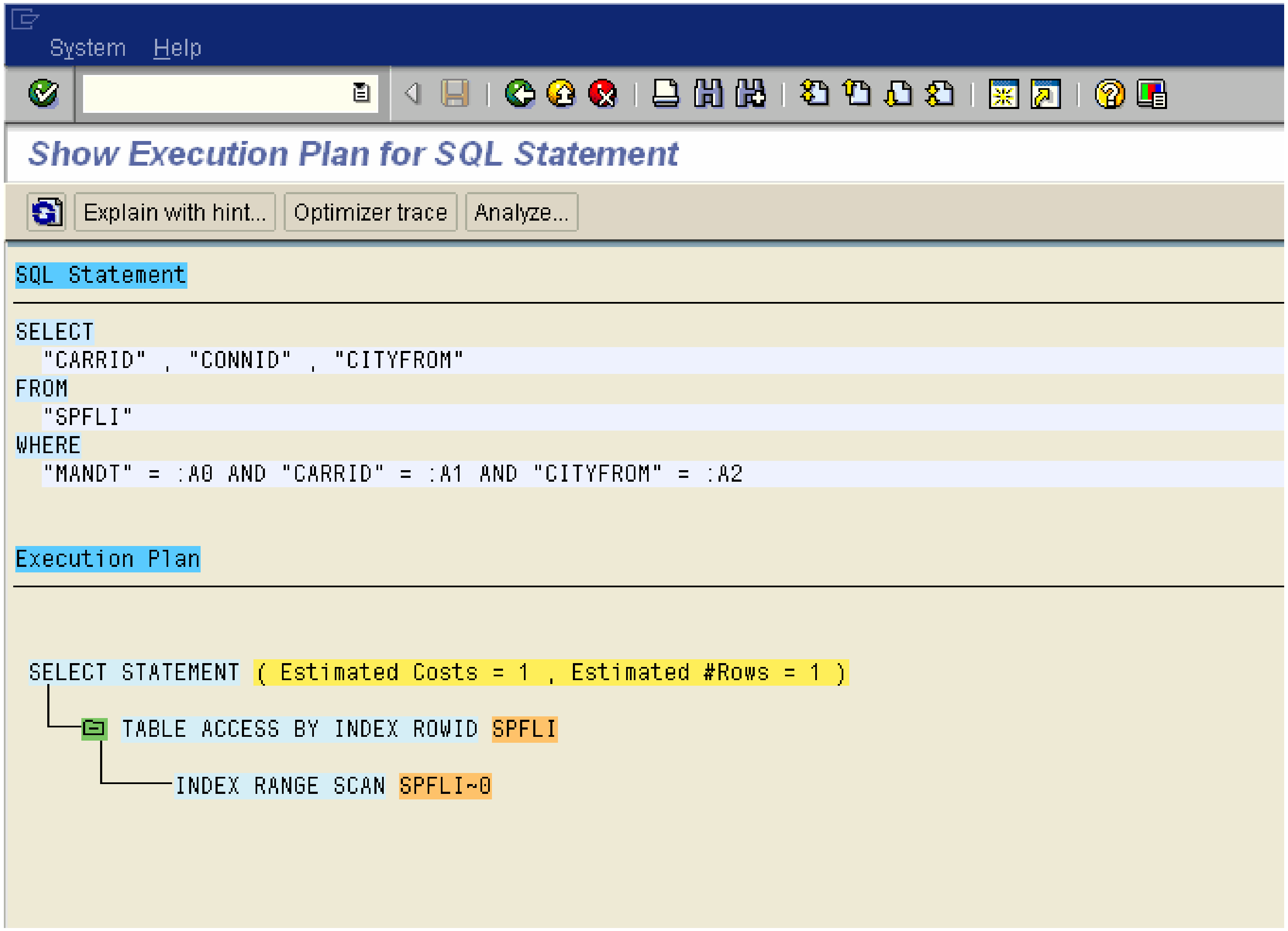


Basic SQL list - sorted by S

Transaction SE38	PID = 26706	P type DIA	Client 000	User = KURKA

Duration	Object	Op.	RJu	RC	Statement
21.555	PROGDIR	REOPEN		0	SELECT WHERE "NAME" = 'AKSQL001' AND "STATE" = 'I' AND ROWNUM <= 1
80.152	PROGDIR	FETCH	0	1403	
242	PROGDIR	REOPEN		0	SELECT WHERE "NAME" = 'AKSQL001' AND "STATE" = 'A' AND ROWNUM <= 1
3.310	PROGDIR	FETCH	1	0	
320	EUF4VALUES	REOPEN		0	SELECT WHERE "RELID" = 'EU' AND "UNAME" = 'KURKA' AND "OBJECT" = 'P' AND "SRTF2" >= 0 OR
6.311	EUF4VALUES	FETCH	1	1403	
3.782	EUF4VALUES	REEXEC	1	0	UPDATE SET "FELD1" = ' ' , "FELD2" = ' ' , "CLUSTR" = 210 , "CLUSTD" = <LRAW> WHERE "REL
2.509	EUF4VALUES	REEXEC	0	0	DELETE WHERE "RELID" = 'EU' AND "UNAME" = 'KURKA' AND "OBJECT" = 'P' AND "SRTF2" = 1
28.479		EXECSTA	0	0	COMMIT WORK ON CONNECTION 0
242	TRDIR	REOPEN		0	SELECT WHERE "NAME" = 'AKSQL001' AND ROWNUM <= 1
3.779	TRDIR	FETCH	1	0	
247	PROGDIR	REOPEN		0	SELECT WHERE "NAME" = 'AKSQL001' AND "STATE" = 'I' AND ROWNUM <= 1
5.834	PROGDIR	FETCH	0	1403	
240	PROGDIR	REOPEN		0	SELECT WHERE "NAME" = 'AKSQL001' AND "STATE" = 'A' AND ROWNUM <= 1
3.119	PROGDIR	FETCH	1	0	
218	DWWASYNC	REOPEN		0	SELECT WHERE "UNAME" = 'KURKA' AND ROWNUM <= 1
2.383	DWWASYNC	FETCH	1	0	
250	DWINACTIV	REOPEN		0	SELECT WHERE "UNAME" = 'KURKA'
5.666	DWINACTIV	FETCH	0	1403	
280	DWINACTIV	REOPEN		0	SELECT WHERE "OBJECT" = 'REPS' AND "OBJ_NAME" = 'AKSQL001'
4.848	DWINACTIV	FETCH	0	1403	
2.570	SPFLI	PREPARE		0	SELECT WHERE "MANDT" = :A0 AND "CARRID" = :A1 AND "CITYFROM" = :A2
262	SPFLI	OPEN		0	SELECT WHERE "MANDT" = '000' AND "CARRID" = 'LH' AND "CITYFROM" = 'FRANKFURT'
8.363	SPFLI	FETCH	6	1403	
317	D345T	REOPEN		0	SELECT WHERE "PROGNAME" = 'SAPMSSY0' AND "SPRACHE" = 'E' AND "BLOCKNR" >= 1 ORDER BY 1 ,
74.232	D345T	FETCH	2	1403	

Execution Plan



SQL Statement

```
SELECT
   "CARRID" , "CONNID" , "CITYFROM"
FROM
   "SPFLI"
WHERE
   "MANDT" = :A0 AND "CARRID" = :A1 AND "CITYFROM" = :A2
```

Execution Plan

```
 SELECT STATEMENT ( Estimated Costs = 1 , Estimated #Rows = 1 )
   |
   |--- TABLE ACCESS BY INDEX ROWID SPFLI
          |
          |------ INDEX RANGE SCAN SPFLI~0
```

Other Tools

There are other tools you should also regard when performance problems show up:

Transaction SE30: The ABAP Profiler

Transaction DB01: Lockwait Situations

Transaction ST02: Buffer Statistics

Transaction ST04: Database Performance Analysis

Agenda

WAS and Database Architecture

ABAP Open SQL Overview

How to Identify Expensive SQL

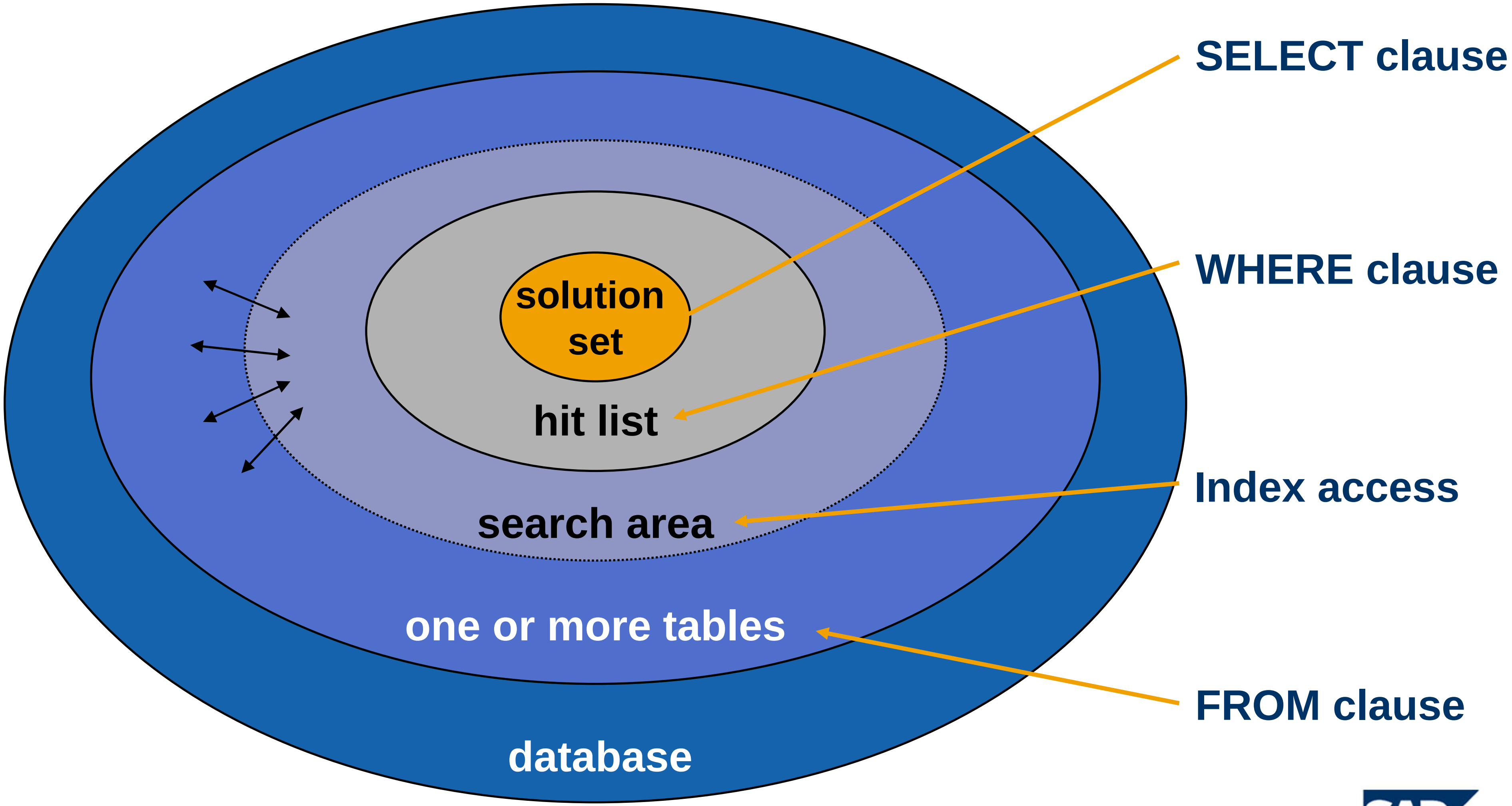
Rules for Better SQL Programming

- Theory and
- Hands-On

Summary

SELECT Statement Overview

```
SELECT p~carrid p~connid cityfrom fldate INTO (crid, cnid, from, date)
  FROM spfli AS p JOIN sflight AS f
         ON p~carrid = f~carrid AND p~connid = f~connid)
  WHERE cityfrom <> 'ROME' AND fldate LIKE '200211%'.
```



Rule 1: Keep the Hit List Small

Keep the hit list small!

Keep the Hit List Small

Use a WHERE clause whenever possible

```
SELECT * FROM sflight
  INTO xflight.
    CHECK xflight-carrid = 'LH '.
    CHECK xflight-connid = '0300'.
    CHECK xflight-fldate(4) = '2002'.
    WRITE: / xflight-fldate.
ENDSELECT.
```

```
SELECT * FROM sflight
  INTO xflight
  WHERE carrid = 'LH '  AND
        connid = '0300' AND
        fldate LIKE '2002%'.
    WRITE: / xflight-fldate.
ENDSELECT.
```

Keep the Hit List Small

Try to describe the full search condition

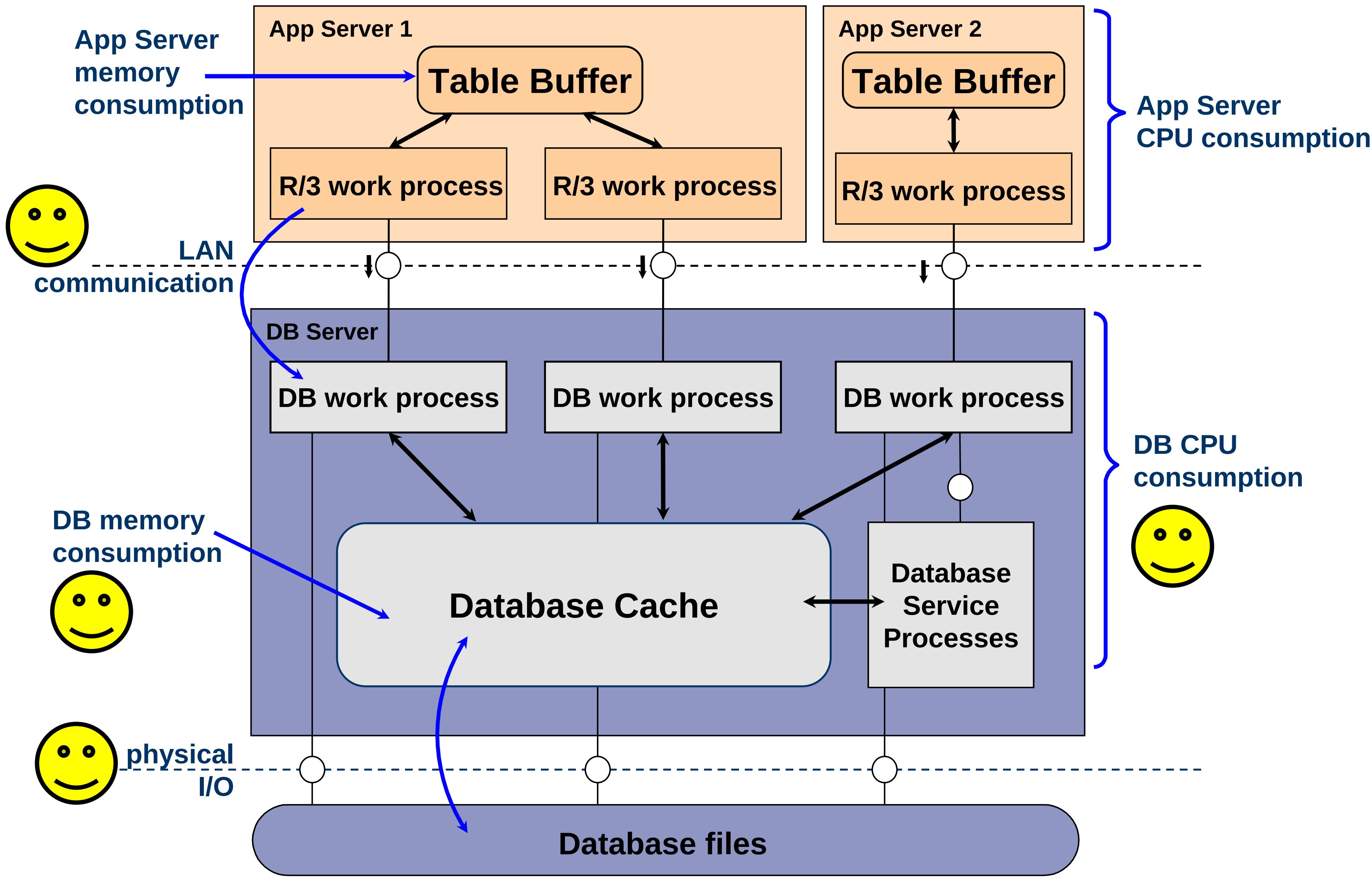
```
SELECT * FROM sflight
  INTO xflight
  WHERE carrid = 'LH ' AND connid = '0300'.
    CHECK xflight-fldate(4) = '2002'.
    WRITE: / xflight-fldate.
ENDSELECT.
```

✗

```
SELECT * FROM sflight
  INTO xflight
  WHERE carrid = 'LH '  AND
        connid = '0300' AND
        fldate LIKE '2002%'.
    WRITE: / xflight-fldate.
ENDSELECT.
```

✓

Keep the Hit List Small: Effects



Keep the Hit List Small: Exercise

Exercise 1: Keep the Hit List Small

1. Open program zwr3d2w3_1_xx for editing.
   (xx = the number of your group)
2. Optimize the SELECT-statements in form "version2":
   Substitute the CHECK-conditions by specifying the desired rows in a WHERE-clause.
3. Run the program to see the effect of your optimization.

Rule 2: Minimize the Amount of Transferred Data

Minimize the
amount of data
transferred between
the database and the application server!

Minimize the Amount of Transferred Data

Use a field list instead of SELECT *

```
SELECT * FROM sflight
  INTO xflight
  WHERE carrid = 'LH '  AND
        connid = '0300' AND
        fldate LIKE '2002%'.
    WRITE: / xflight-fldate.
ENDSELECT.
```

✗

```
SELECT fldate FROM sflight
  INTO (xflight-fldate)
  WHERE carrid = 'LH '  AND
        connid = '0300' AND
        fldate LIKE '2002%'.
    WRITE: / xflight-fldate.
ENDSELECT.
```

✓

Minimize the Amount of Transferred Data: Exercises

Exercise 2: Specify a SELECT-List

1. **Open program zwr3d2w3 _2_xx for editing.**
   (xx = the number of your group)
2. **Optimize the SELECT-statements in form "version3":**
   Select only the columns needed in the subroutine.
3. **Run the program to see the effect of your optimization.**

Minimize the Amount of Transferred Data

Apply UP TO *n* ROWS for a top-*n* solution set

```
SELECT id name discount
  FROM scustom
  INTO (xid, xname, xdiscount)
  WHERE custtype = 'B'
  ORDER BY discount.
    IF sy-dbcnt > 10. EXIT. ENDIF.
    WRITE: / xid, xname, xdiscount.
ENDSELECT.
```

✗

```
SELECT id name discount
  FROM scustom UP TO 10 ROWS
  INTO (xid, xname, xdiscount)
  WHERE custtype = 'B'
  ORDER BY discount.
    WRITE: / xid, xname, xdiscount.
ENDSELECT.
```

✓

Minimize the Amount of Transferred Data

Use the UPDATE ... SET Statement

```
SELECT * FROM sflight
    INTO xflight
    WHERE carrid ='LH '.
  xflight-seatsocc = xflight-seatsocc + 1.
  UPDATE sflight FROM xflight.
ENDSELECT.
```

✗

```
UPDATE sflight
  SET seatsocc = seatsocc + 1
  WHERE carrid = 'LH '.
```

✓

Minimize the Amount of Transferred Data

Use aggregate functions

```
sum = 0.
SELECT seatsocc
  FROM sflight INTO xseatsocc
  WHERE fldate LIKE '2002%'.
    sum = sum + xseatsocc.
ENDSELECT.
WRITE: / sum.
```

✗

```
SELECT SINGLE SUM( seatsocc )
  FROM sflight INTO sum
  WHERE fldate LIKE '2002%'.
WRITE: / sum.
```

✓

Minimize the Amount of Transferred Data

Apply the HAVING clause

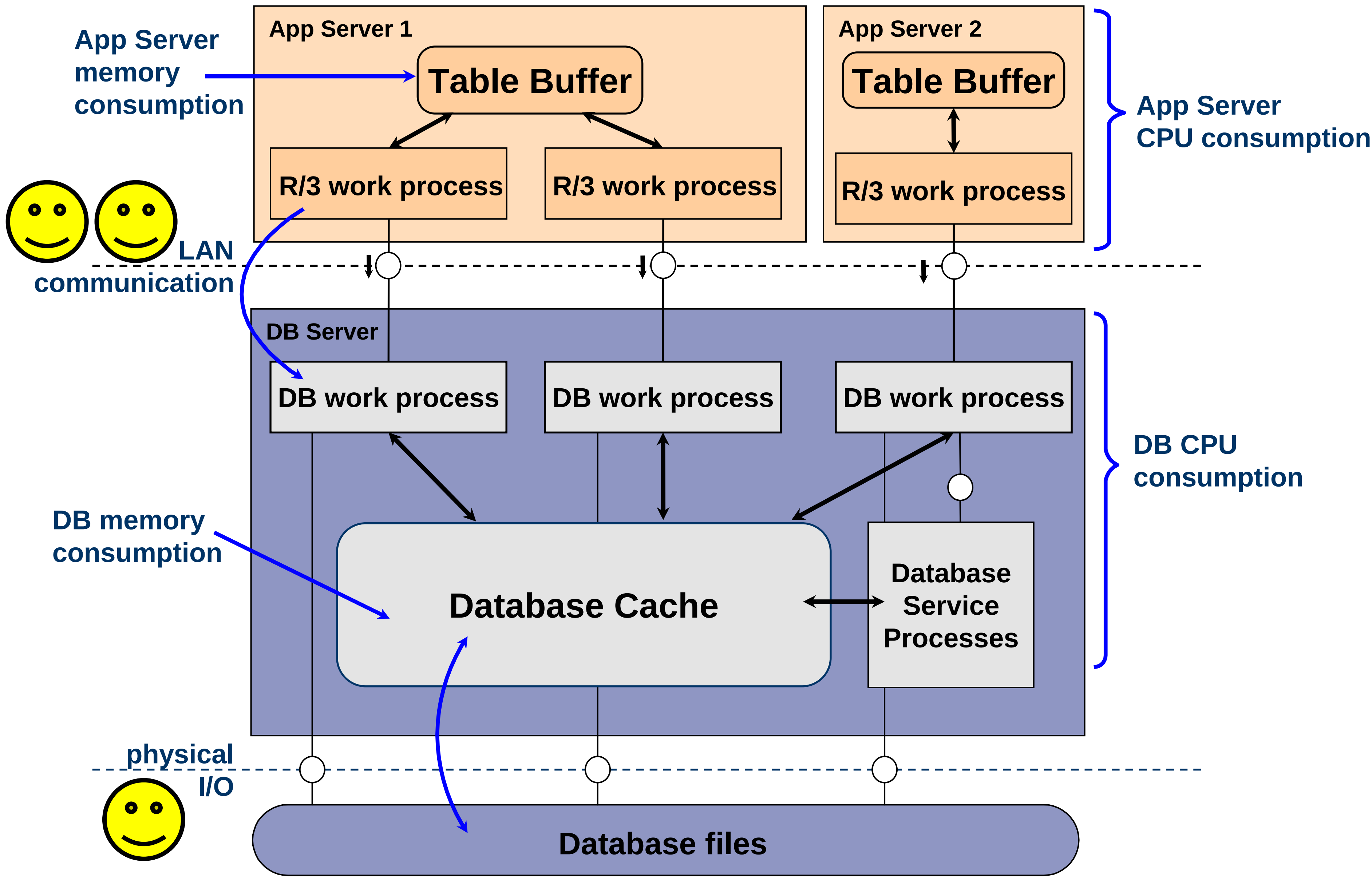
```
SELECT carrid connid fldate MAX( luggweight )
    INTO (xcarrid, xconnid, xfldate, max)
    FROM sbook
    GROUP BY carrid connid fldate.
  CHECK max  > 20.
  WRITE: / xcarrid, xconnid, xfldate, max.
ENDSELECT.
```

✗

```
SELECT carrid connid fldate MAX( luggweight )
    INTO (xcarrid, xconnid, xfldate, max)
    FROM sbook
    GROUP BY carrid connid fldate
    HAVING MAX( luggweight ) > 20.
  WRITE: / xcarrid, xconnid, xfldate, max.
ENDSELECT.
```

✓

Minimize the Amount of Transferred Data: Effects



Minimize the Amount of Transferred Data: Exercises

Exercise 3: Use Aggregate Functions

1. Open program zwr3d2w3 _3_xx for editing.
   (xx = the number of your group)
2. Optimize the SELECT-statements in form "version4":
   Have the database calculate the sum in the inner loop.
3. Run the program to see the effect of your optimization.

Rule 3: Keep the Number of Round Trips Small

Keep the number of
round trips
between the database
and the application server
small!

Keep the Number of Round Trips Small

Use high-speed array operations with UPDATE, INSERT, DELETE, MODIFY

```
LOOP AT itab INTO wa.
  INSERT INTO sbook VALUES wa.
ENDLOOP.
```

✗

```
INSERT sbook FROM TABLE itab.
```

✓

Keep the Number of Round Trips Small

Apply the INNER JOIN
Avoid nested SELECT-ENDSELECT loops

```
SELECT * FROM sflight INTO xflight WHERE planetype = '727-200'.
  SELECT * FROM sbook INTO xbook
      WHERE carrid = xflight-carrid AND
            connid = xflight-connid AND
            fldate = xsflight-fldate.
    WRITE: / xflight-carrid, xflight-connid, xbook-bookid.
  ENDSELECT.
ENDSELECT.
```

✗

```
SELECT f~carrid f~connid b~bookid
    INTO (xcarrid, xconnid, xbookid)
    FROM sflight AS f INNER JOIN sbook AS b
         ON f~carrid = b~carrid AND
            f~connid = b~connid AND
            f~fldate = b~fldate
    WHERE planetype = '727-200'.
  WRITE: / xcarrid, xconnid, xbookid.
ENDSELECT.
```

✓

Keep the Number of Round Trips Small

Apply the OUTER JOIN

```
SELECT * FROM sflight INTO xflight WHERE planetype = '727-200'.
  SELECT * FROM sbook INTO xbook
      WHERE carrid = xflight-carrid
        AND connid = xflight-connid
        AND fldate = xflight-fldate.
    WRITE: / xflight-carrid, xflight-connid, xflight-fldate,
             xbook-bookid.
  ENDSELECT.
  IF sy-dbcnt = 0.
    CLEAR xbook-bookid.
    WRITE: / xflight-carrid, xflight-connid, xflight-fldate,
             xbook-bookid.
  ENDIF.
ENDSELECT.
```

✗

```
SELECT f~carrid f~connid f~fldate b~bookid
    INTO (xcarrid, xconnid, xfldate, xbookid)
    FROM sflight AS f LEFT OUTER JOIN sbook AS b
         ON  f~carrid = b~carrid AND f~connid = b~connid
             AND f~fldate = b~fldate.
    WHERE planetype = '727-200'.
  WRITE: / xcarrid, xconnid, xfldate, xbookid.
ENDSELECT.
```

✓

Keep the Number of Round Trips Small

Use subqueries

```
SELECT carrid connid MAX( seatsocc )
  FROM sflight
  INTO (xcarrid, xconnid, max)
  GROUP BY carrid connid
  ORDER BY carrid connid.
    SELECT fldate FROM sflight
        INTO yfldate
        WHERE carrid   = xcarrid AND
              connid   = xconnid AND
              seatsocc = max
        ORDER BY fldate.
      WRITE: / xcarrid, xconnid, yfldate.
    ENDSELECT.
ENDSELECT.
```

✗

```
SELECT carrid connid fldate
    FROM sflight AS f
    INTO (xcarrid, xconnid, xfldate)
    WHERE seatsocc IN
     ( SELECT MAX( seatsocc ) FROM sflight
       WHERE carrid = f~carrid AND connid = f~connid )
    ORDER BY carrid connid fldate.
  WRITE: xcarrid, xconnid, xfldate.
ENDSELECT.
```

✓

Keep the Number of Round Trips Small

For frequently used INNER JOINs, you can create a database view in the ABAP Dictionary

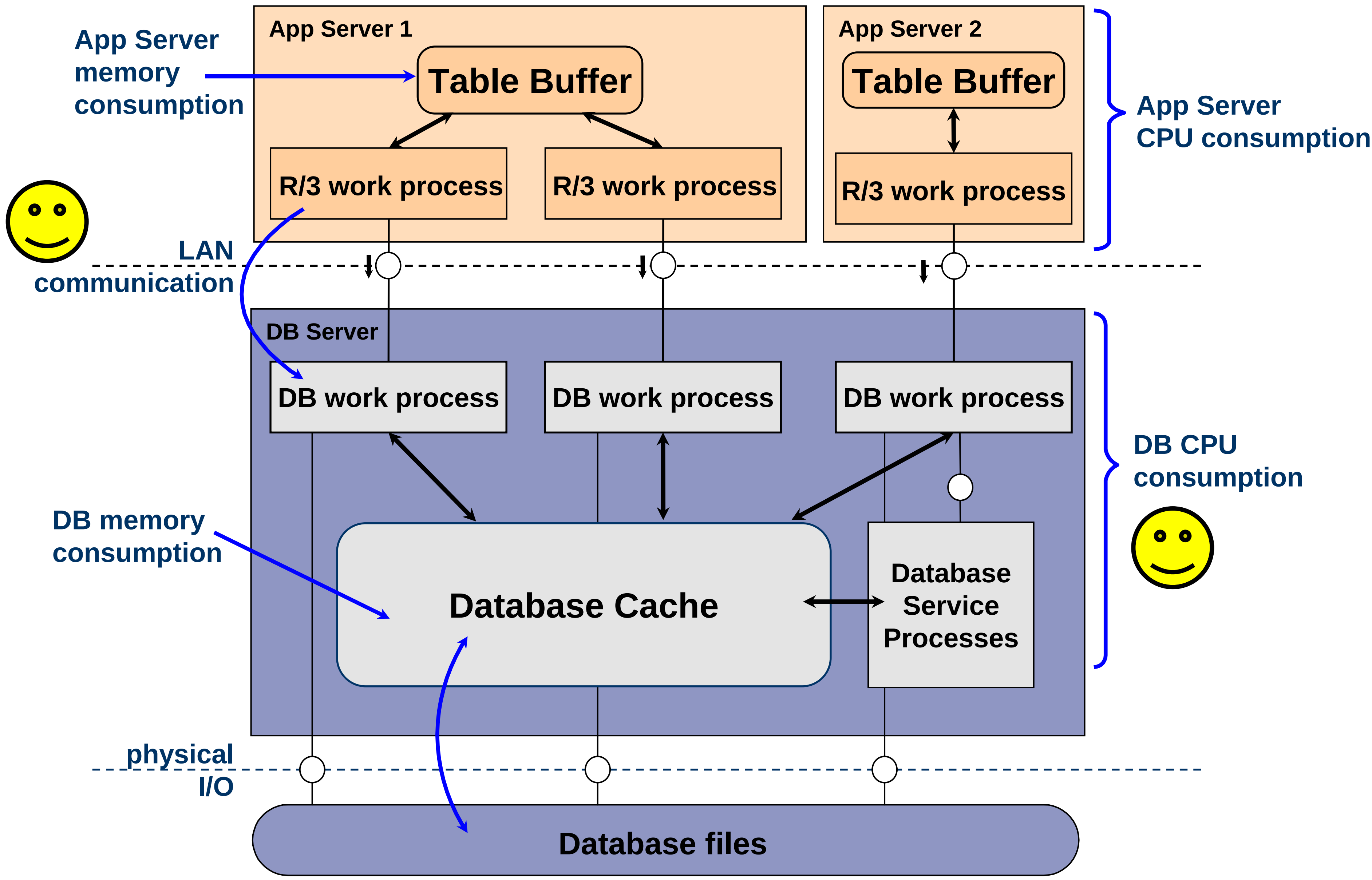
```
SELECT f~carrid f~connid b~bookid
    INTO (xcarrid, xconnid, xbookid)
    FROM sflight AS f INNER JOIN sbook AS b
         ON  f~carrid = b~carrid AND f~connid = b~connid
             AND f~fldate = b~fldate.
  WRITE: / xcarrid, xconnid, xbookid.
ENDSELECT.
```

✗

```
SELECT carrid connid bookid
    INTO (xcarrid, xconnid, xbookid)
    FROM sflightbook.
  WRITE: / xcarrid, xconnid, xbookid.
ENDSELECT.
```

✓

Keep the Number of Round Trips Small: Effects



Keep the Number of Round Trips Small: Exercise

Exercise 4: Use an Inner Join

1. Open program zwr3d2w3 _4_xx for editing.
   (xx = the number of your group)
2. Optimize the select-statements in form "version5":
   Replace the nested SELECT-ENDSELECT-loops by an inner join.
3. Run the program to see the effect of your optimization.

Rule 4: Keep the Cost of the Search Down

Keep the
cost of the search
down!

Keep the Cost of the Search Down

Specify the WHERE clause to keep the number of searches down and create suitable indices if necessary

```
SELECT bookid
    FROM sbook INTO xflight
    WHERE orderdate = '20020304'.
  WRITE: / xbookid.
ENDSELECT.
```

✗

```
SELECT bookid
    FROM sbook INTO xbookid
    WHERE carrid = 'LH '  AND
        connid = '0300' AND
        fldate = '20020304'.
  WRITE: / xbookid.
ENDSELECT.
```

✓

Reasonable Index Design

- **Keep in mind, which indices are defined**
- **Place fields that are effective in the selection process at the beginning**
- **The following fields are *not* effective in the selection process: MANDT, BUKRS, GJAHR.**
- **Create small indices**
- **Avoid overlaps (create disjunctive indices)**
- **Up to 4 indices in each table generally are not critical**

Keep the Cost of the Search Down

Make sure that the first *n* fields of the designated index are stated with EQ within the WHERE clause

```
SELECT * FROM sflight
    INTO xflight
    WHERE carrid = 'LH '  AND
          fldate LIKE '2002%'.
  WRITE: / xflight-fldate.
ENDSELECT.
```

✗

```
SELECT * FROM sflight
    INTO xflight
    WHERE carrid = 'LH '  AND
          connid = '0300' AND
          fldate LIKE '2002%'.
  WRITE: / xflight-fldate.
ENDSELECT.
```

✓

Keep the Cost of the Search Down

Replace the inner OR with an IN operator

```
SELECT * FROM sflight
    INTO xflight
    WHERE carrid = 'LH ' AND
          (connid = '0300' OR connid = '0302') AND
           fldate LIKE '2002%'.
  WRITE: / xflight-fldate.
ENDSELECT.
```

✗

```
SELECT * FROM sflight
    INTO xflight
    WHERE carrid = 'LH ' AND
          connid IN ('0300', '0302') AND
          fldate LIKE '2002%'.
  WRITE: / xflight-fldate.
ENDSELECT.
```

✓

Keep the Cost of the Search Down

You cannot process NOT operators in SELECT using an index

```
SELECT * FROM sflight
    INTO xflight
    WHERE carrid <> 'LH ' AND
          connid = '0300'.
  WRITE: / xflight-fldate.
ENDSELECT.
```

✗

```
SELECT * FROM sflight
    INTO xflight
    WHERE carrid IN ('AA ', 'QM ') AND
          connid = '0300'.
  WRITE: / xflight-fldate.
ENDSELECT.
```

✓

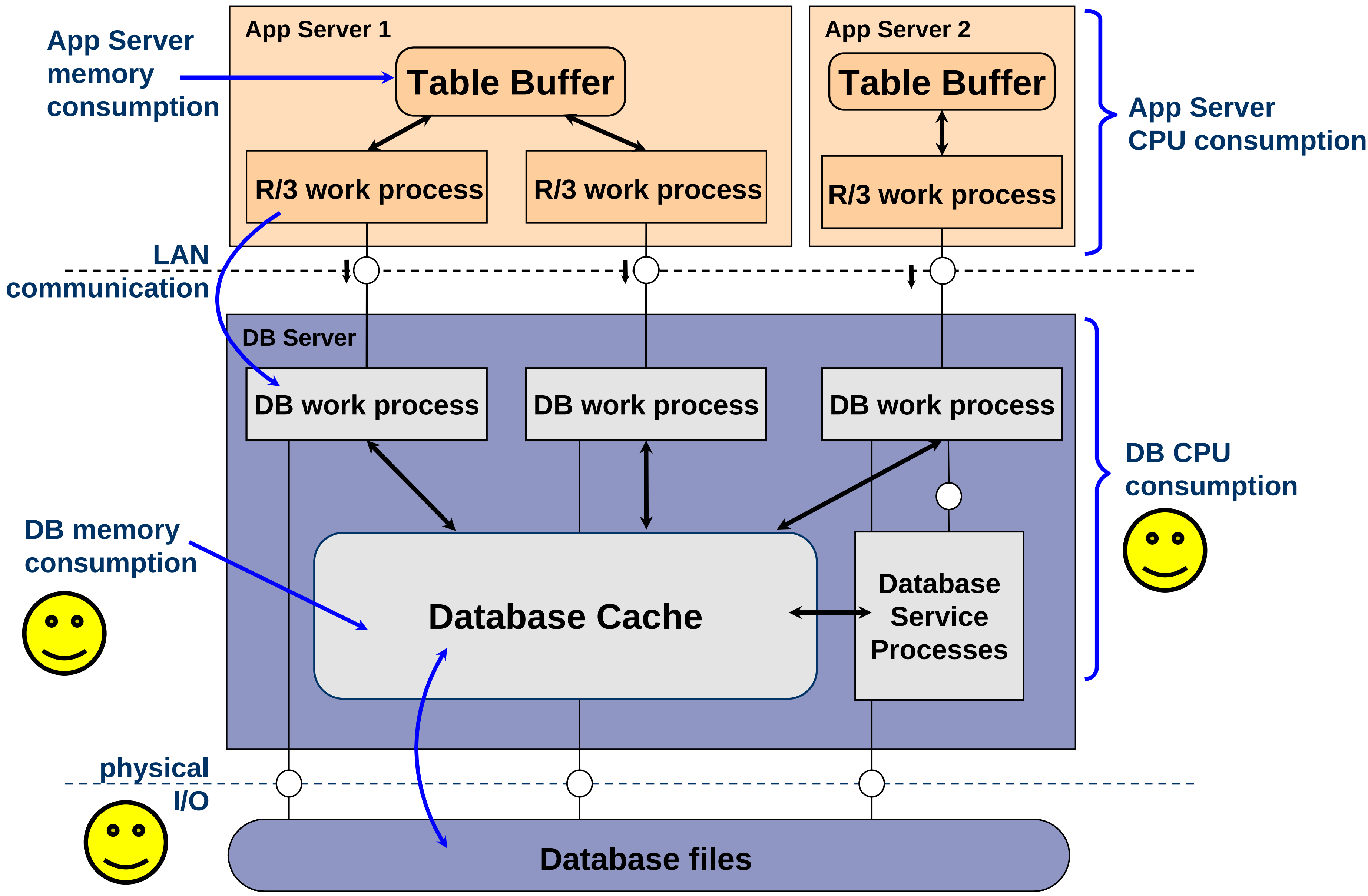
Keep the Cost of the Search Down

Think about optimizer hints if the optimizer fails to find a sound execution plan

```
SELECT carrid connid cityfrom
    FROM spfli INTO (xcarrid, xconnid, xcityfrom)
    WHERE carrid = 'LH ' AND cityfrom = 'FRANKFURT'.
  WRITE: / xcarrid, xconnid, xcityfrom.
ENDSELECT.
```

```
SELECT carrid connid cityfrom
    FROM spfli INTO (xcarrid, xconnid, xcityfrom)
    WHERE carrid = 'LH ' AND cityfrom = 'FRANKFURT'
    %_HINTS ORACLE 'INDEX("SPFLI" "SPFLI~001")'.
  WRITE: / xcarrid, xconnid, xcityfrom.
ENDSELECT.
```

Keep the Cost of the Search Down: Effects



Rule 5: Remove the Load From the Database

Remove the
load
from the
database!

Remove the Load From the Database

Check whether a table meets the criteria for Table Buffering

```
SELECT SINGLE * FROM scarr
  INTO xcarr
  WHERE carrid = 'LH '.
```

Criteria for Table Buffering

When to apply table buffering

- Frequently read
- Relatively small
- Deferred visibility of changes is acceptable

When to avoid table buffering

- Heavily changed
- Contents must always be up-to-date

Statements that Bypass the Table Buffer

- **SELECT .... DISTINCT**
- **SELECT .... COUNT, SUM, AVG, MIN, MAX**
- **SELECT .... ORDER BY f1 ... fn**
- **SELECT .... GROUP BY / HAVING**
- **SELECT .... FOR UPDATE**
- **SELECT .... JOIN**
- **WHERE clause contains IS NULL statement**
- **WHERE clause contains subquery**

- **SELECT ... BYPASSING BUFFER**

Remove the Load From the Database

Avoid reading the same data again and again

```
SELECT SINGLE * FROM scarr
  INTO xcarr
  WHERE carrid = 'LH '.
...
SELECT SINGLE * FROM scarr
  INTO zcarr
  WHERE carrid = 'LH '.
...
```

```
SELECT SINGLE * FROM scarr
  INTO xcarr
  WHERE carrid = 'LH '.
...
zcarr = xcarr.
...
```

Remove the Load From the Database

Check whether a SELECT is really needed before an UPDATE is made

```
SELECT SINGLE * FROM sflight
  INTO xflight
  WHERE carrid = 'LH '  AND
        connid = '0300' AND
        fldate = '20021204'.
xflight-seatsocc = 1.
UPDATE sflight FROM xflight.
```

✗

```
UPDATE sflight
  SET seatsocc = 1
  WHERE carrid = 'LH '  AND
        connid = '0300' AND
        fldate = '20021204'.
```

✓

Remove the Load From the Database

Avoid the ORDER BY clause if the desired sorting doesn’t correspond to the index used

```
SELECT p~airpfrom p~airpto f~fldate p~deptime
    INTO xflight
    FROM spfli AS p INNER JOIN sflight AS f
      ON     p~carrid    = f~carrid
        AND p~connid    = f~connid
    WHERE p~carrid = 'LH '
    ORDER BY p~airpfrom p~airpto f~fldate p~deptime.
  WRITE: / xflight-airpfrom, xflight-airpto,
           xflight-fldate, xflight-deptime.
ENDSELECT.
```

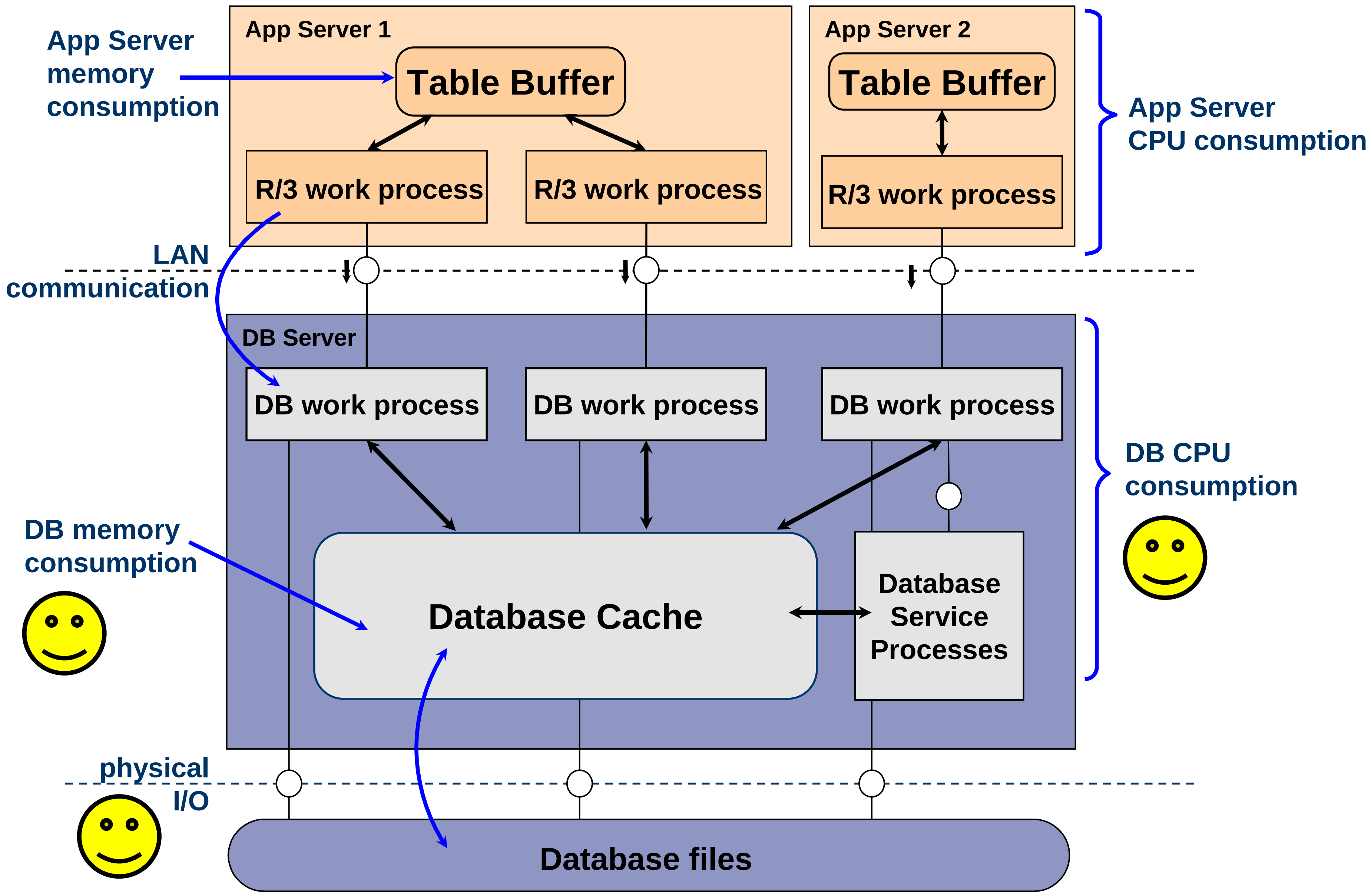
✗

```
SELECT p~airpfrom p~airpto f~fldate p~deptime
  INTO TABLE flights
  FROM spfli AS p INNER JOIN sflight AS f
     ON     p~carrid    = f~carrid
       AND p~connid    = f~connid
  WHERE p~carrid = 'LH '.
SORT flights BY airpfrom airpto fldate deptime.

LOOP AT flights INTO xflight.
  WRITE: / xflight-airpfrom, xflight-airpto,
           xflight-fldate, xflight-deptime.
ENDLOOP.
```

✓

Remove the Load From the Database: Effects



Golden Rule: Think and Experiment

Think and experiment!

Think and Experiment

- Take recommendations as rules of thumb rather than laws
- Some of the rules unveil their benefits only if you use tables of a certain minimum capacity
- Some of the goals of the rules are even inconsistent
- Recommendations hold true for all SAP-supported DB systems

Agenda

Summary

There is just one database server

Buffers and indices

Check their usage via SQL Trace

Try to stick to the presented rules:

- Small hit list
- Minimize transfers
- Minimize number of round trips
- Narrow your search
- Minimize database load

Further Information

Related Workshops at TechEd 2002

Analyzing Performance with the Code Inspector
Nov. 12, 16:15-18:15

Performance Analysis in a Nutshell
Nov. 13, 8:15-12:15

Traps and Pitfalls in ABAP
Nov. 12, 13:45-15:45
Nov. 15, 10:30-12:30

ABAP for Power Users
Nov. 14, 14:00-18:00
Nov. 15, 8:15-12:15

Questions?

Q&A

Feedback

Please complete your session evaluation and drop it in the box on your way out.

Be courteous — deposit your trash, and do not take the handouts for the following session.

Thank You

The SAP TechEd ’02 New Orleans Team

Copyright 2002 SAP AG. All Rights Reserved

THE BEST-RUN BUSINESSES RUN SAP

Copyright 2002 SAP AG. Alle Rechte vorbehalten

THE BEST-RUN BUSINESSES RUN SAP