Firebird OLTP Test

Pavel Zotov, Firebird QA

THANK YOU!

ABOUT THIS TEST

- Emulates work of real-life app (car service)
  - Settings for init pop., warm-up & measure.
- Does not require 3rd party utilities.
- Workload modes: small, medium, heavy.
- Main purpose: get maximal performance.
- Performance report auto creating.
  - Log every unit run and its result.
- Test home: www.firebirdtest.com (to be published)
- Author: Pavel Zotov, p519446@yandex.ru

WHY THIS TEST

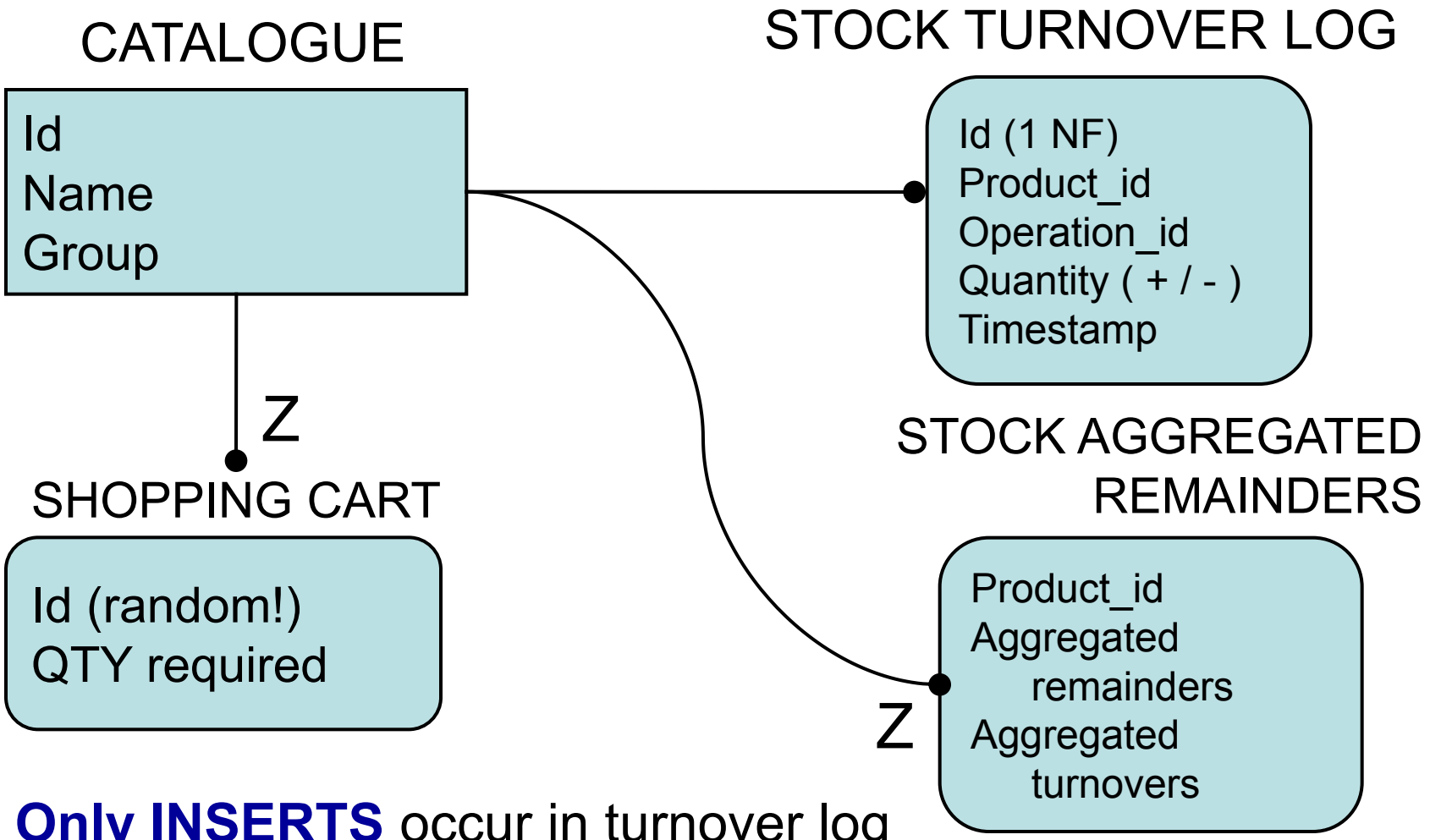
- Stress-test of Firebird stability
- Logic as in real-life, workload much harder
- Compare performance:
  - * “hardware-1” vs “hardware-2”
  - * Firebird 3.0 vs 2.5
  - * SS vs SC vs CS
  - * database settings: page_size, FW, etc
- Create client-side app for distribution as example (planning).

MODEL: INTRO

Main entities:

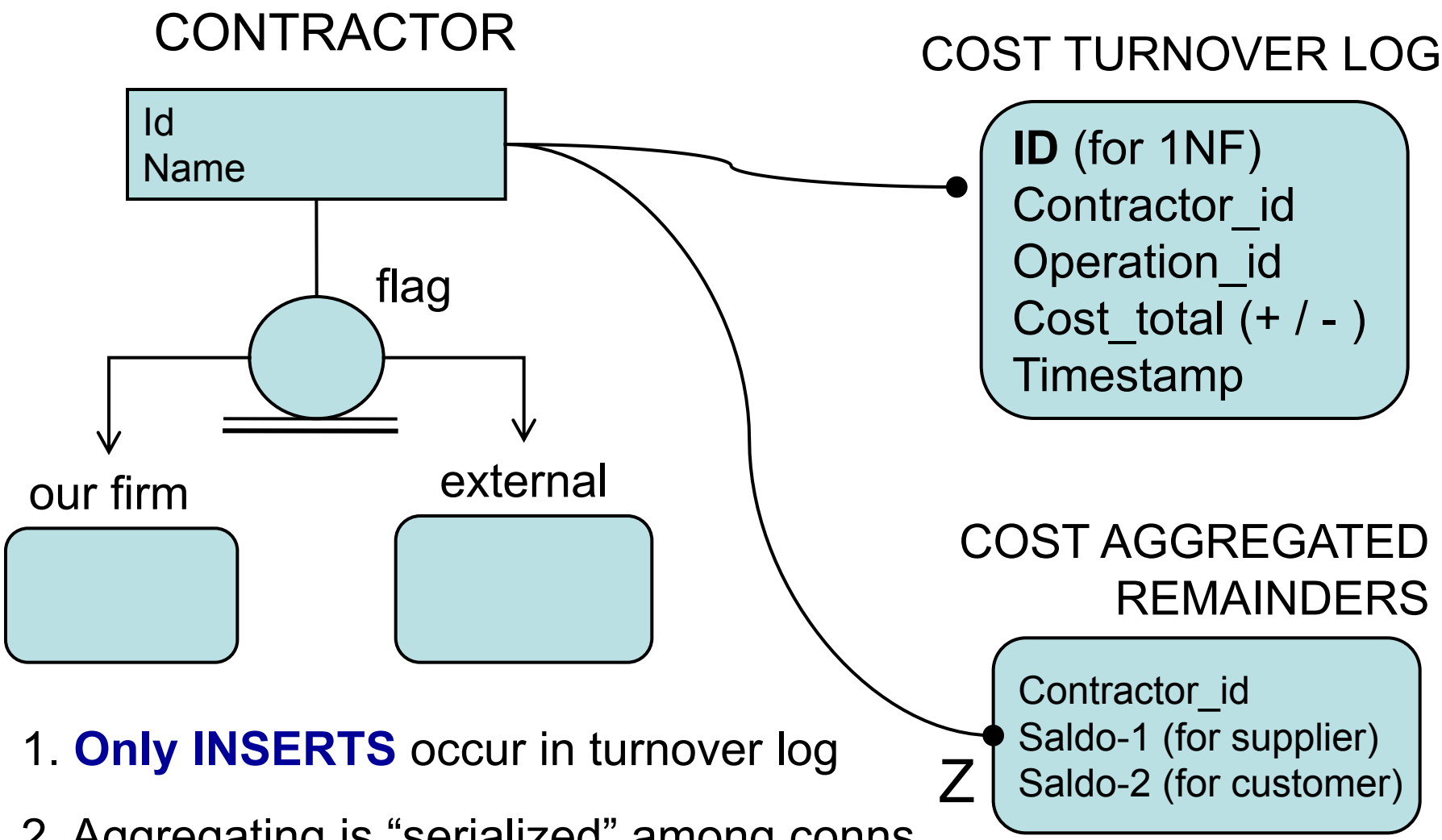
- Catalogue of products & shopping cart
- Contractor
- Document header
- Document line
- Turnover log
- Aggregated remainder
- FIFO distribution: source & target

MODEL: STOCK ENTITIES



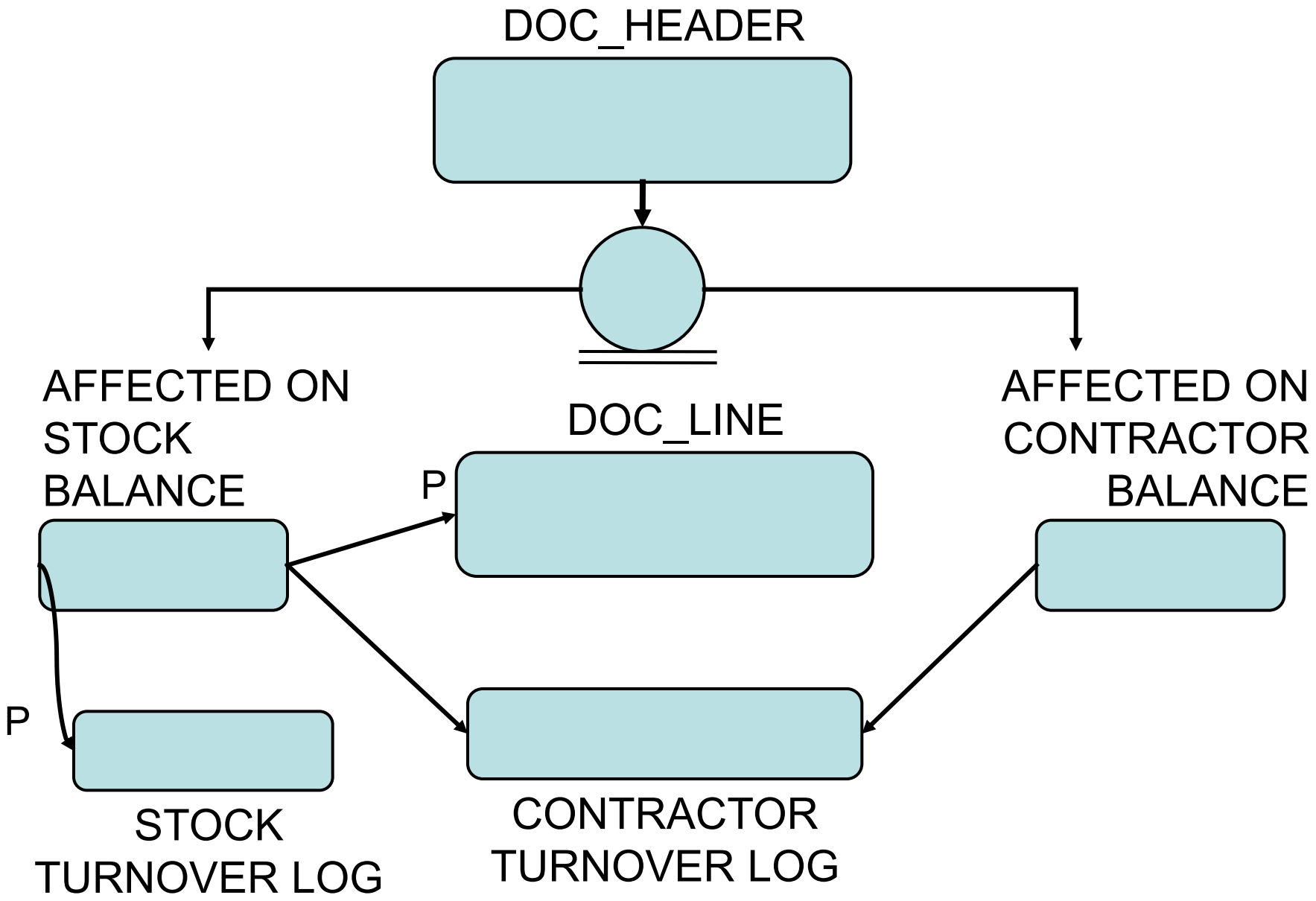
- 1. **Only INSERTS** occur in turnover log
- 2. Aggregating is “serialized” with high frequency

MODEL: CONTRACTORS



1. **Only INSERTS** occur in turnover log
2. Aggregating is “serialized” among conns.
3. “Our firm” => NO reserve after shipping

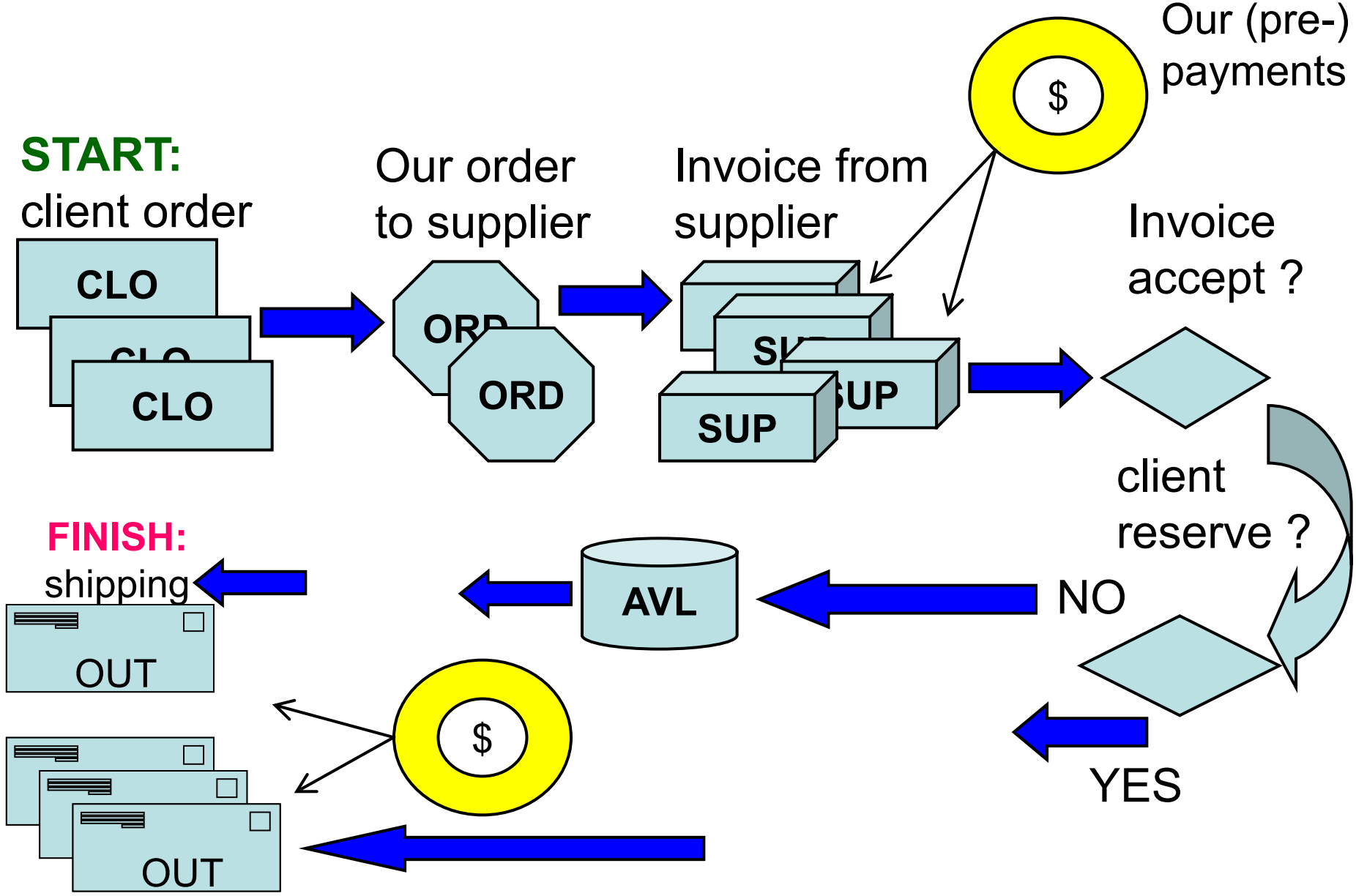
MODEL: DOCS & TURNOVERS



DATABASE SCHEMA

- Documents flow (operations)
- Turnovers and balances
- Effect from operations
- Producer - consumer logic

DOCUMENTS FLOW



TURNOVERS AND BALANCES

STOCK BALANCES:

- **CLO**: client order
- **ORD**: order to supplier
- **SUP**: invoice from suppl.
- **AVL**: available
- **RES**: reserve for client

STOCK TURNOVERS:

- **INC**: total incomings
- **OUT**: total outgoings

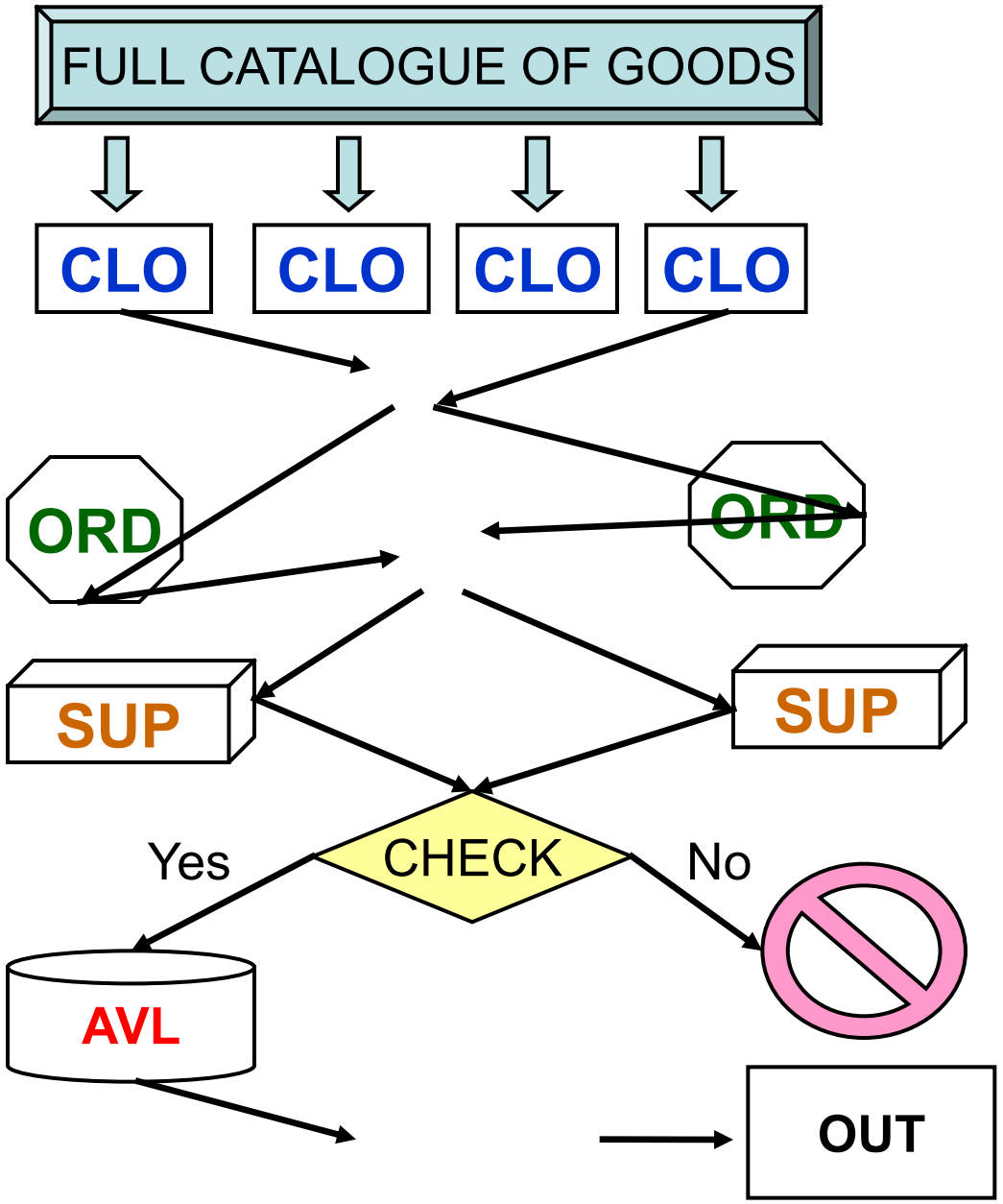
ACCOUNTANT BALANCE:
= **INC** – **OUT**

MONETARY BALANCE & TURNOVER:

- balance of contractor as supplier or as customer:
  is calculated in purchasing or retailing prices
- (pre-) payment to supplier / (pre-)payment from client

CHANGES OF REMAINDERS

- Catalogue -> **Client** order
- Supplier **order**: gather rows from client orders
- Supplier invoice: gather rows from **supplier** orders
- Check and accepting invoice: add its content to **available** remainders
- Search for client orders that still need some goods to be **reserved**
- Create **reserve**
- Products shipping to customers



EFFECT FROM OPERATIONS

BUSINESS OPERATION	CLO	ORD	SUP	AVL	RES	INC	OUT
CUSTOMER ORDER	1						
OUR ORDER TO SUPPLIER	-1	1					
WE GET INVOICE FROM SUPPLIER		-1	1				
WE VERIFIED & ACCEPT INVOICE			-1	1		1	
RESERVE FOR CLIENT				-1	1		
SALE OF PRODUCTS					-1		1

Legend:

CLO = remainder in customer orders

ORD = remainder in orders to supplier

SUP = not-delivered invoices

AVL = ‘on-hand’ remainder

RES = remainder of reserved goods

INC = total incomings

OUT = total outgoings

Producer-Consumer: **why**?

Too many row-level lock conflicts:

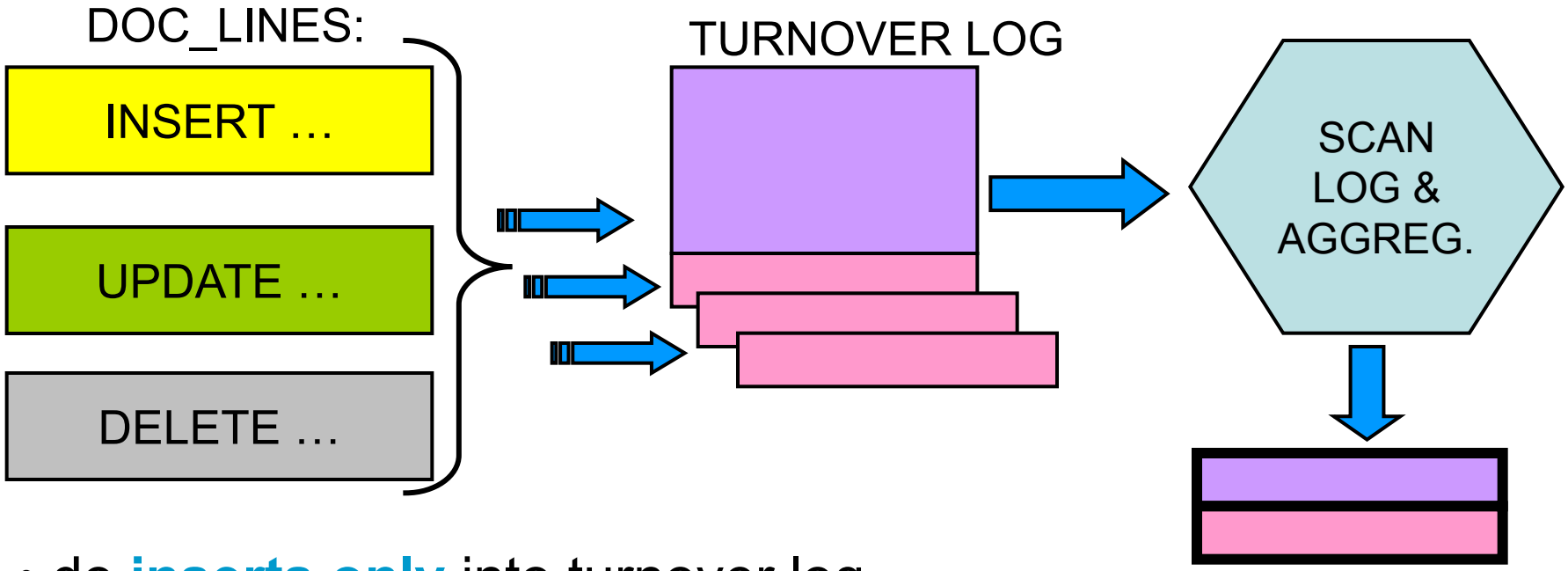
- When need to update **remainder**
- When need to change amount in document **line**
- When need to change total cost in document **header**

Performance impact:

Earlier test versions: approx. 80% of application unit calls failed with lock conflict.
No sense to measure performance in this case.

Producer-Consumer: BALANCES

Attempt to apply same schema as for contractor balances:



- do **inserts only** into turnover log
- only **one attach** runs aggregation
- **clear** turnover log after aggregation finishes

Q: how to provide constraint “REMAINDER >= 0” ?

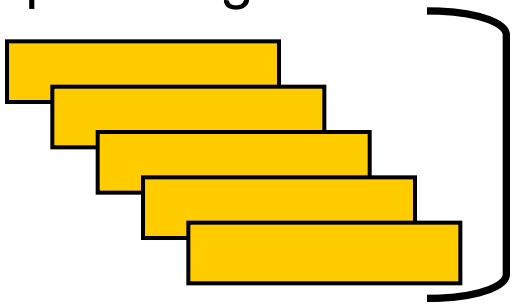
Producer-Consumer: QUANTITIES

Producer:

INSERT INTO
DOC_LINES(…, QTY)
VALUES(…, **5**);

~ “map”

Source for future qty spreading:



ADD **5** ROWS
FOR QTY = 5

Consumers:

Tx1: wants Q=3

Tx2: wants Q=2

Tx3: wants Q=4

id = 125
id = 126
id = 127
id = 128
id = 129

FREE

id = 125
id = 126
id = 127
id = 128
id = 129

LOCKED

Tx1: gets Q=3

Tx2: gets Q=2

Tx3: gets NONE

Don’t allow to take more than **source** can give!

Producer-Consumer: DOC TOTALS

Producer:
DOC_HEADER

INSERT(…, COST)
VALUES(…, **5400**)

~ “map”

Source for future cost spreading:

id=1 $1000	id=4 $1000
id=2 $1000	Id=5 $1000
id=3 $1000	Id=6 $400

Consumers:

Tx1: wants $2100

Tx2: wants $1700

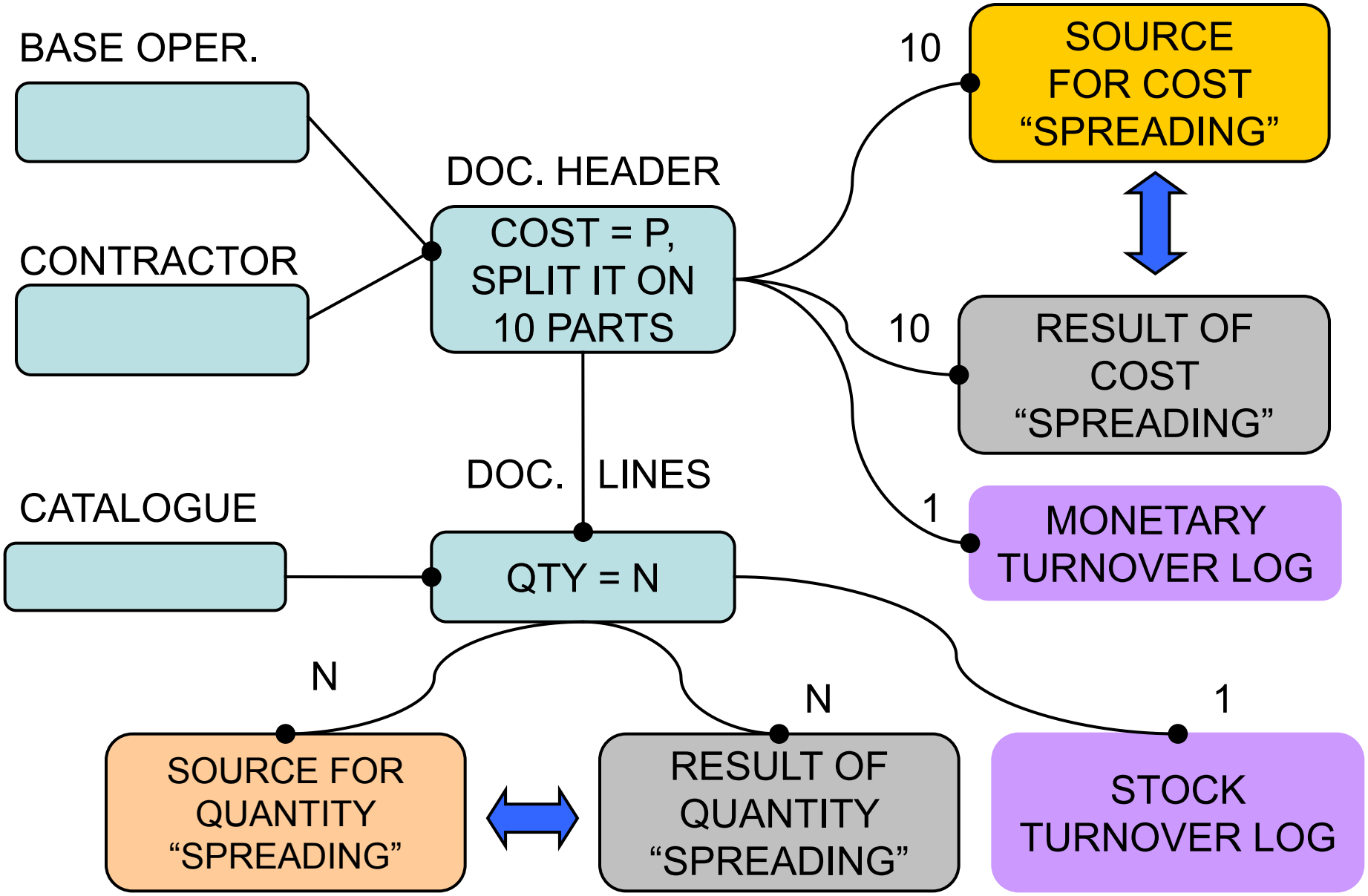
Tx3: wants $800

id=1 $1000	id=1 $1000	→	id=1 $1000
id=2 $1000	id=2 $1000	→	id=2 $1000
id=3 $1000	id=3 $1000	→	$900 $100
id=4 $1000	id=4 $1000	→	id=4 $1000
Id=5 $1000	id=5 $1000	→	$300 $700
Id=6 $400	id=6 $1000	→	$200 $800

FREE

LOCKED

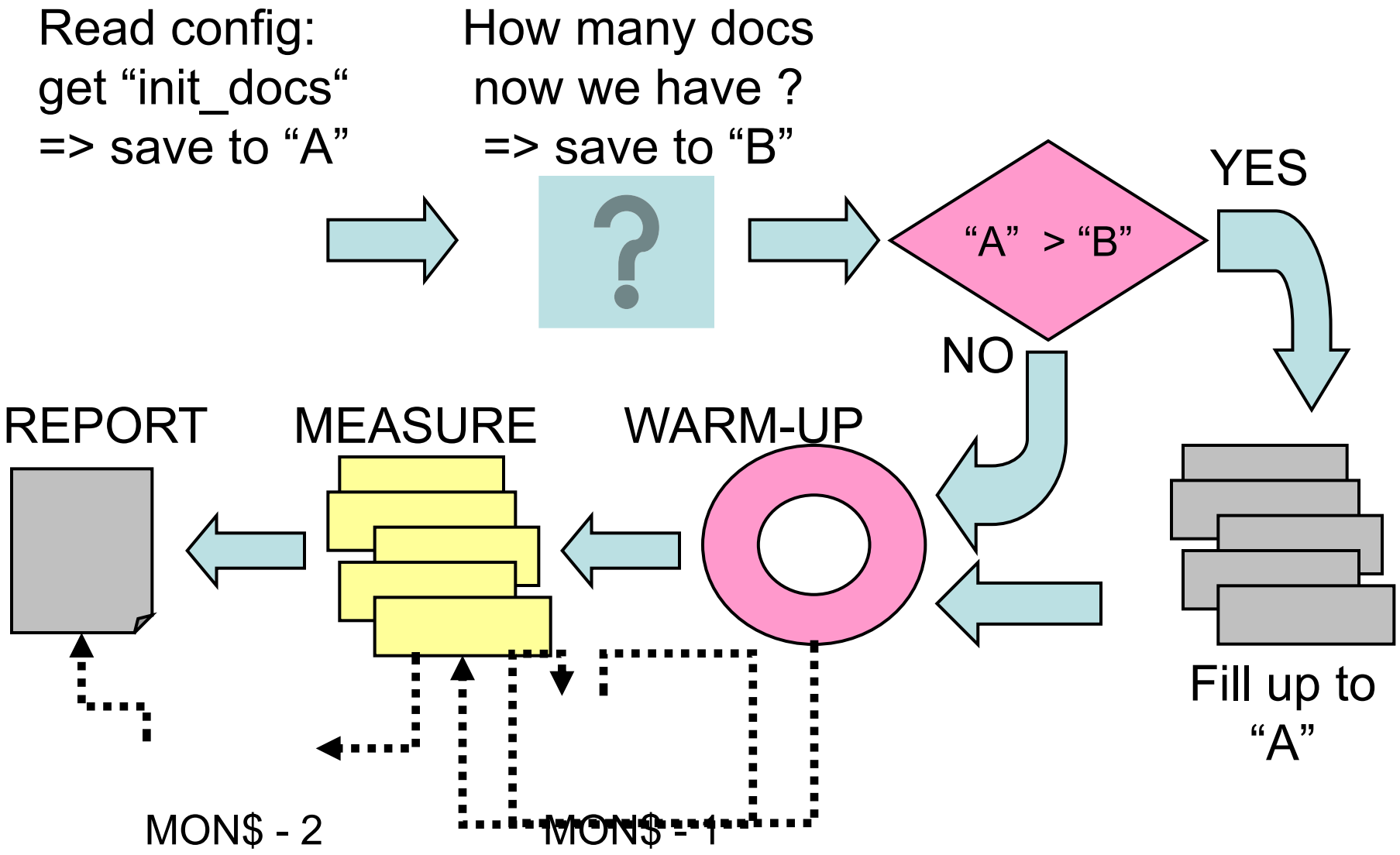
PRODUCER–CONSUMER: OVERALL



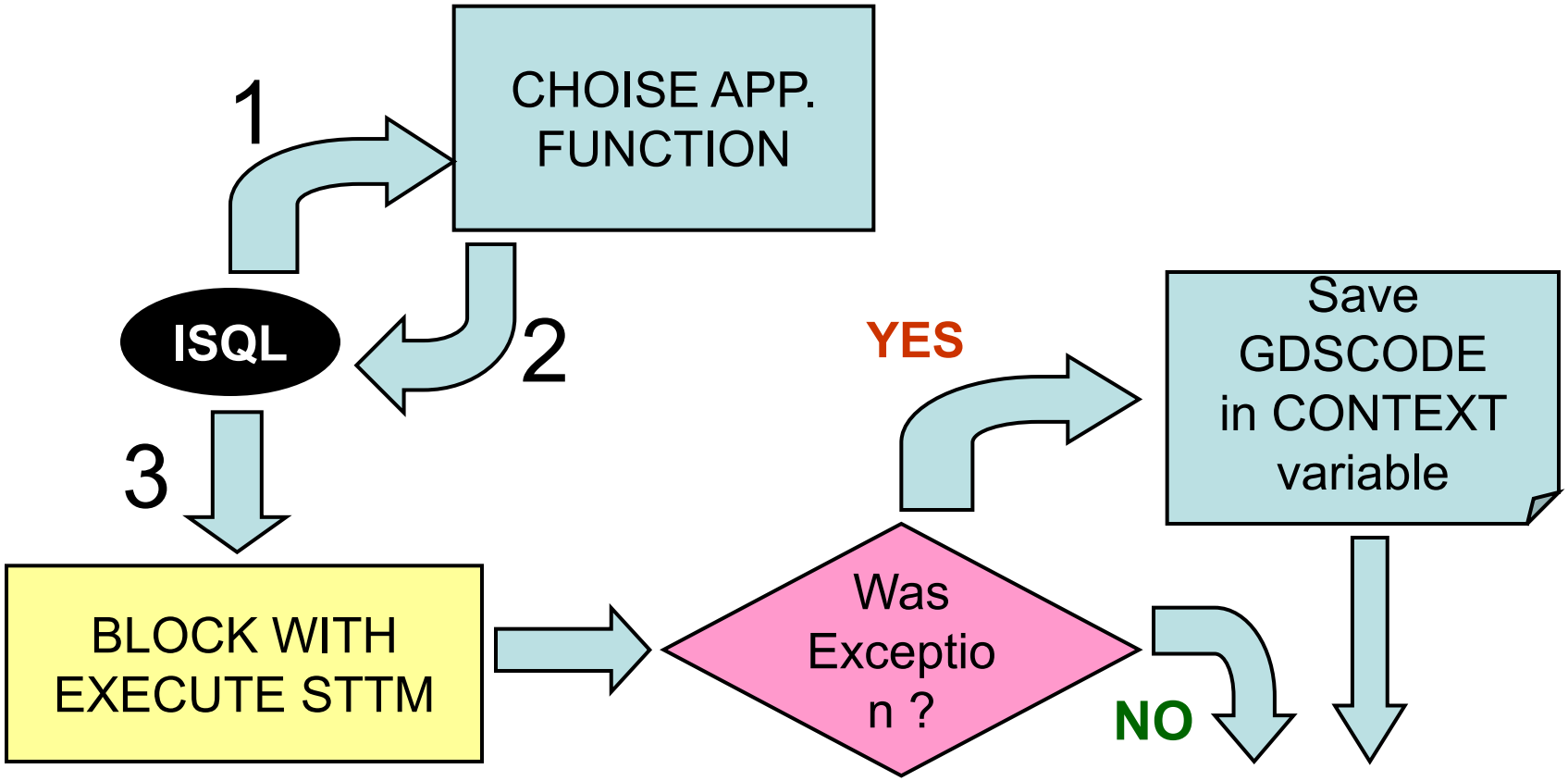
HOW TEST WORKS

- Phases of test run
- Sketch of measurement
- Auto make performance report

PHASES OF TEST RUN

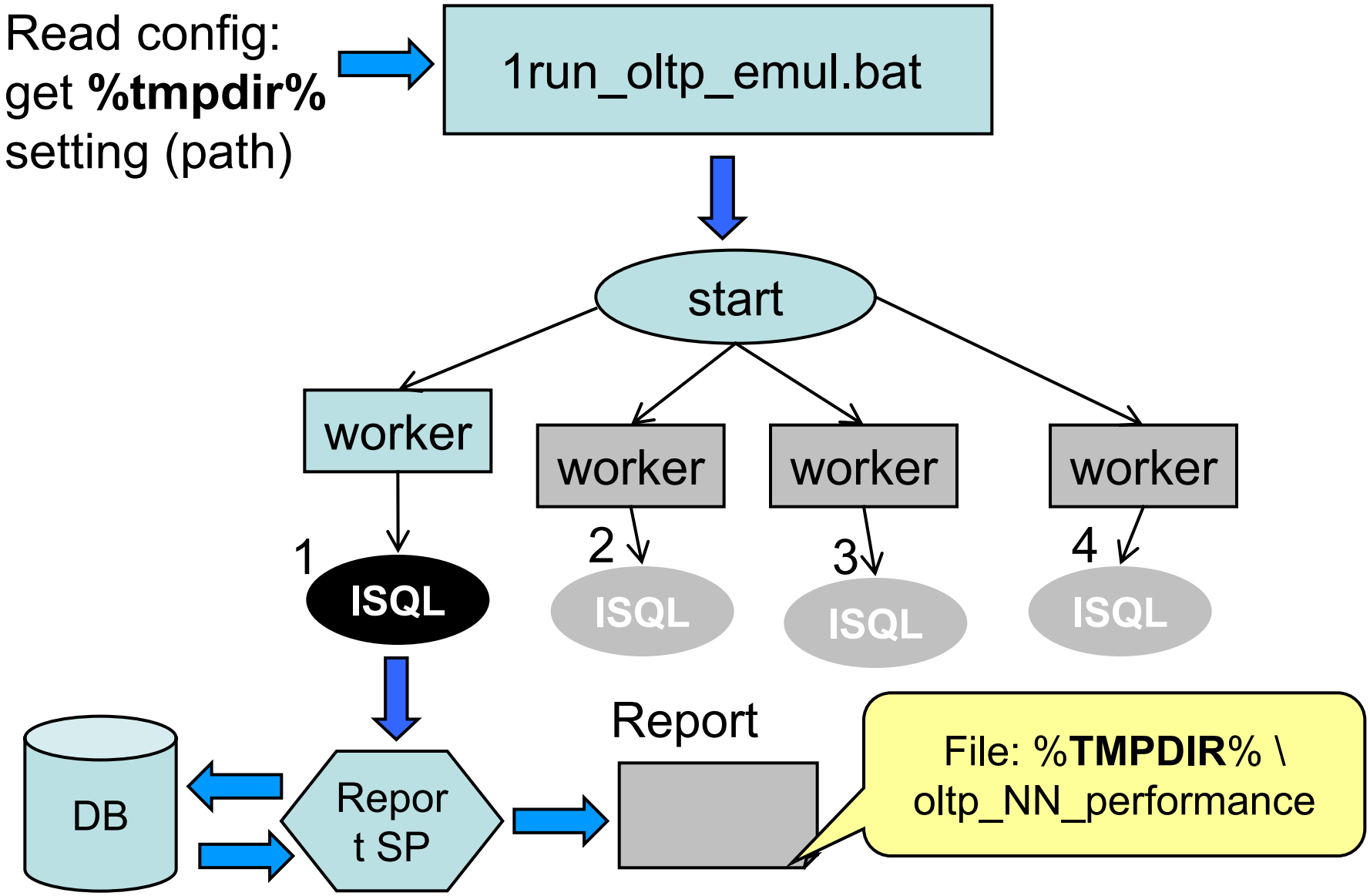


SKETCH OF MEASUREMENT



- Big script with ~300 transactions
- Repeat this after finish (loop in .bat)
- Batch checks whether one need to exit from loop and terminate itself

AUTO PERFORMANCE REPORT



PERFORMANCE: RESULTS

- Performance: how to measure ?
- Results:
  performance overall
  dynamic change of performance
  explanation
- What it was tested ?
  Hardware, Firebird & database settings
- Graphics

PERFORMANCE: IN WHAT “UNITS” ?

Performance rating: P = S / M,

where:

S = number of *successfully* completed actions, when gdscode is NULL

M = durability of workload period, in minutes

PERFORMANCE: REPORTS

Following reports can be created:

- 1) overall;
- 2) dynamic (“how Firebird gets tired”);
- 3) detailed;
- 4) exceptions occurred

WHAT IS WAS TESTED ?

- Server:
  12 core CPU, 2GHz, RAM 32 Gb HDD IBM SCSI
  OS: Linux RHEL, kernel 2.6.39

- Firebird versions: 2.5 SS, 2.5 SC, 3.0 SS, 3.0 SC
- Database settings:
  FW = ON and OFF
  page_size = 8192

Number of attaches: 25, 50, 100, 150

- Initial number of documents: 30000. Database size: ~410 Mb
- Database warm-up time: 10 minutes (3.0), 15 minutes (2.5)
- Measured time: mostly 180 minutes, several times 12 hours

Changes in firebird.conf

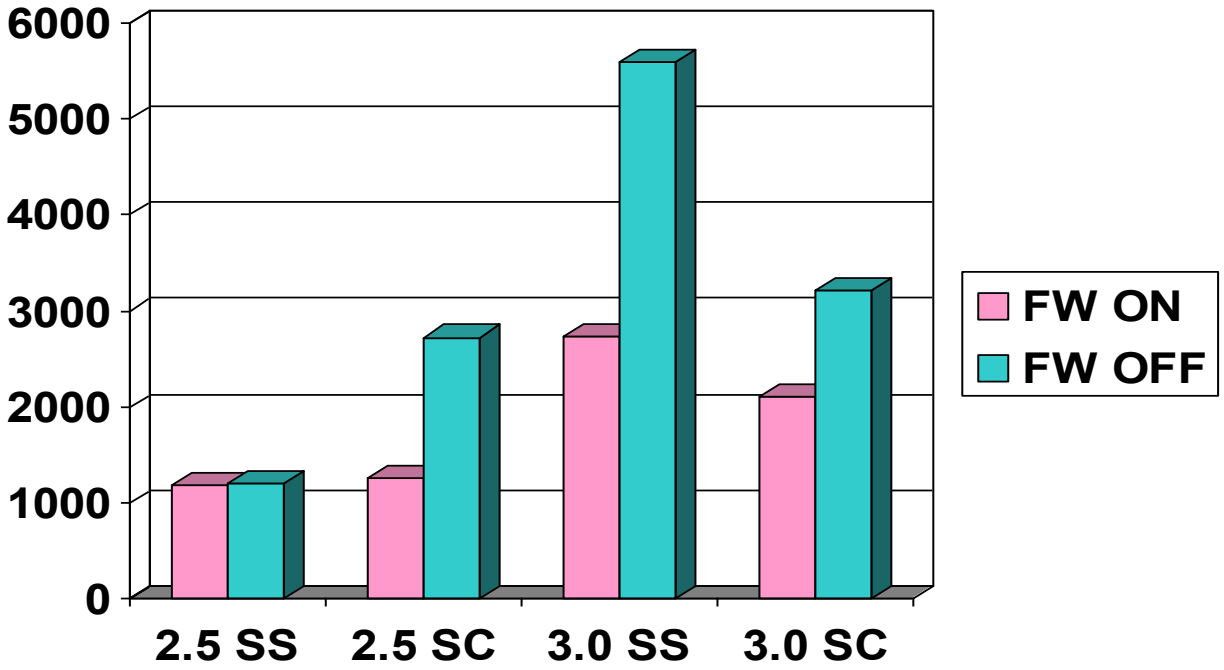
Following parameters need to be changed:

- **ExternalFileAccess = Restrict <path>**
  (place when 'STOPTEST.TXT' will live)

- DefaultDBCachePages
  increase at least to 512 for SC or CS
  increase at least to 65535 for SS
- LockHashSlots
  increase to 22111
- TempCacheLimit
  increase at least to 256M

RESULTS - PERFORMANCE OVERALL

Successful business actions per minute, in average:



Number of attaches: 100

Warm-up time: 10 minutes

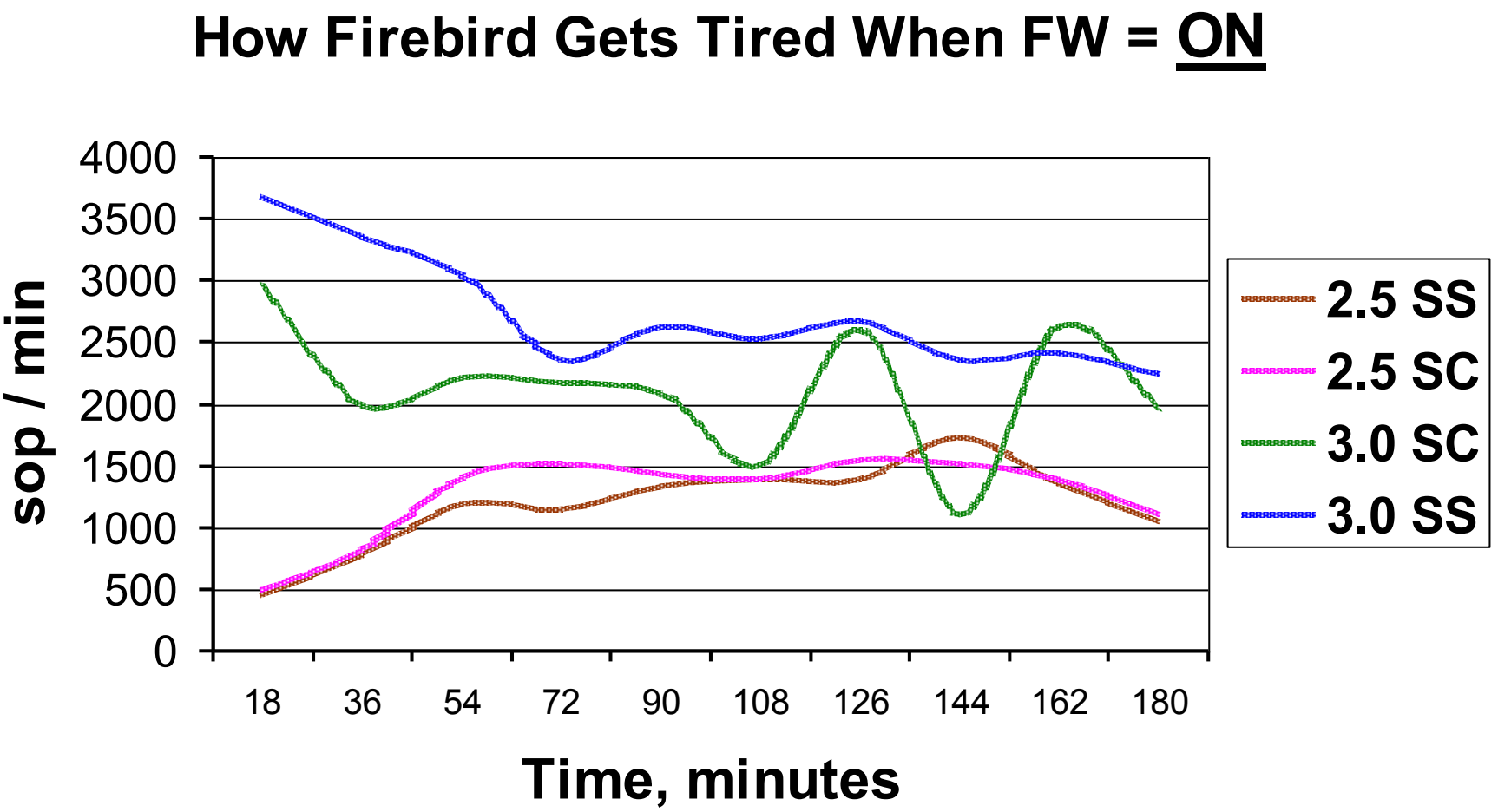
Measurement time: 180 minutes

DefaultDbCachePages = 512K

LockHashSlots = 22111

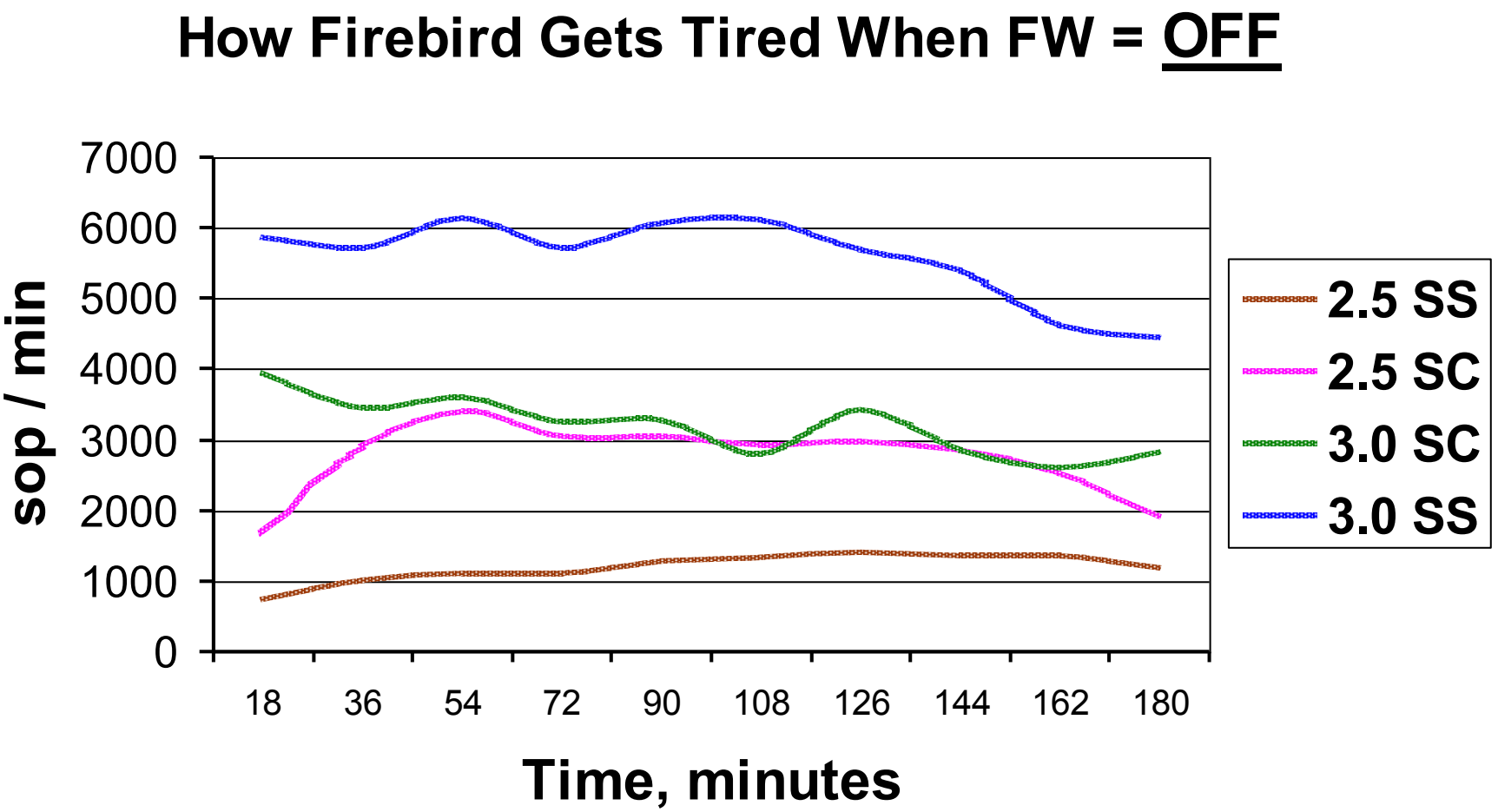
TempCacheLimit = 2 Gb

PERFORMANCE IN TIME, FW = ON



(successful business actions per minute, in average)

PERFORMANCE IN TIME, FW = OFF



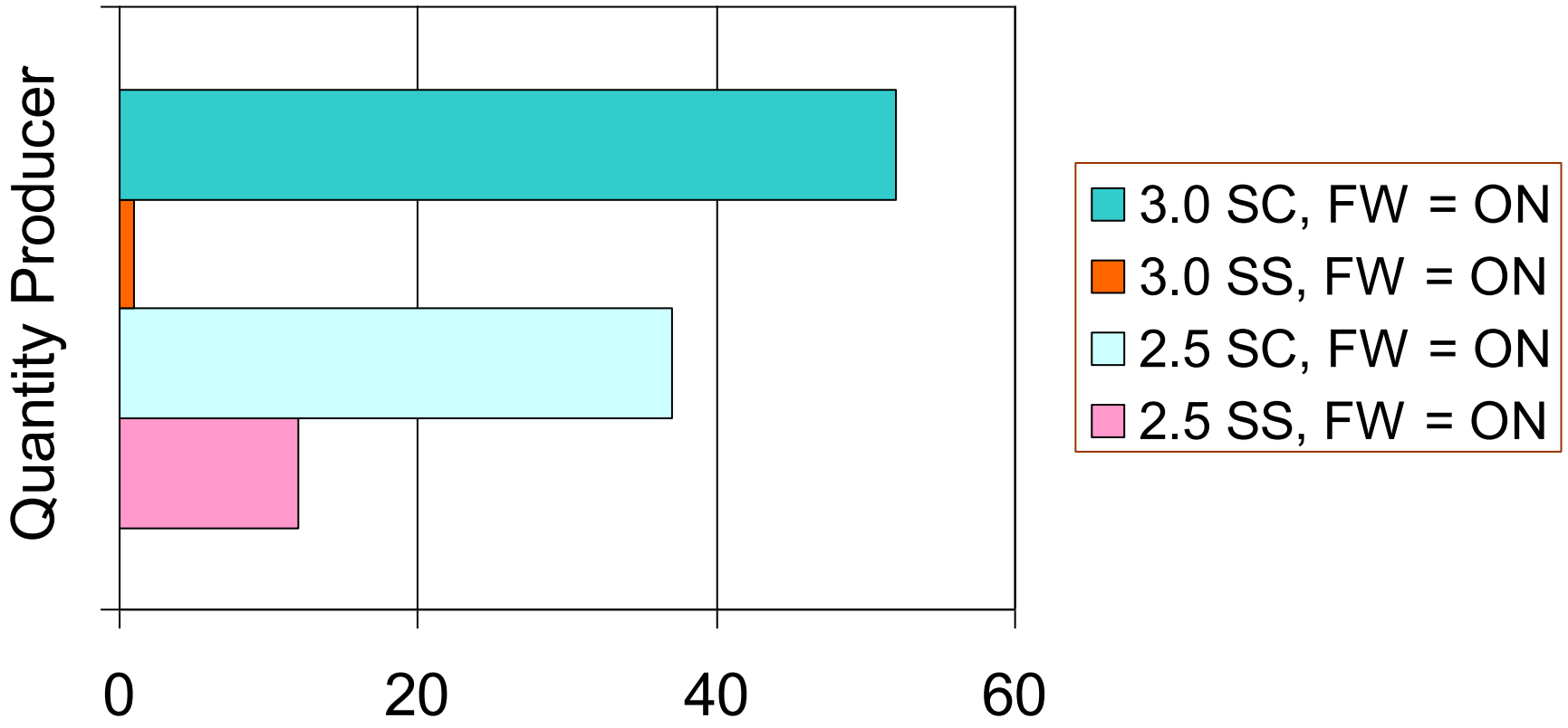
(successful business actions per minute, in average)

WORKLOAD & RECORD VERSIONS, FW = ON (1/2)

Attaches: 100. Warm-up: 10 min. Measure: 180 min.

Table: **producer** of quantity for FIFO distribution

Total Versions / Total Records, %:

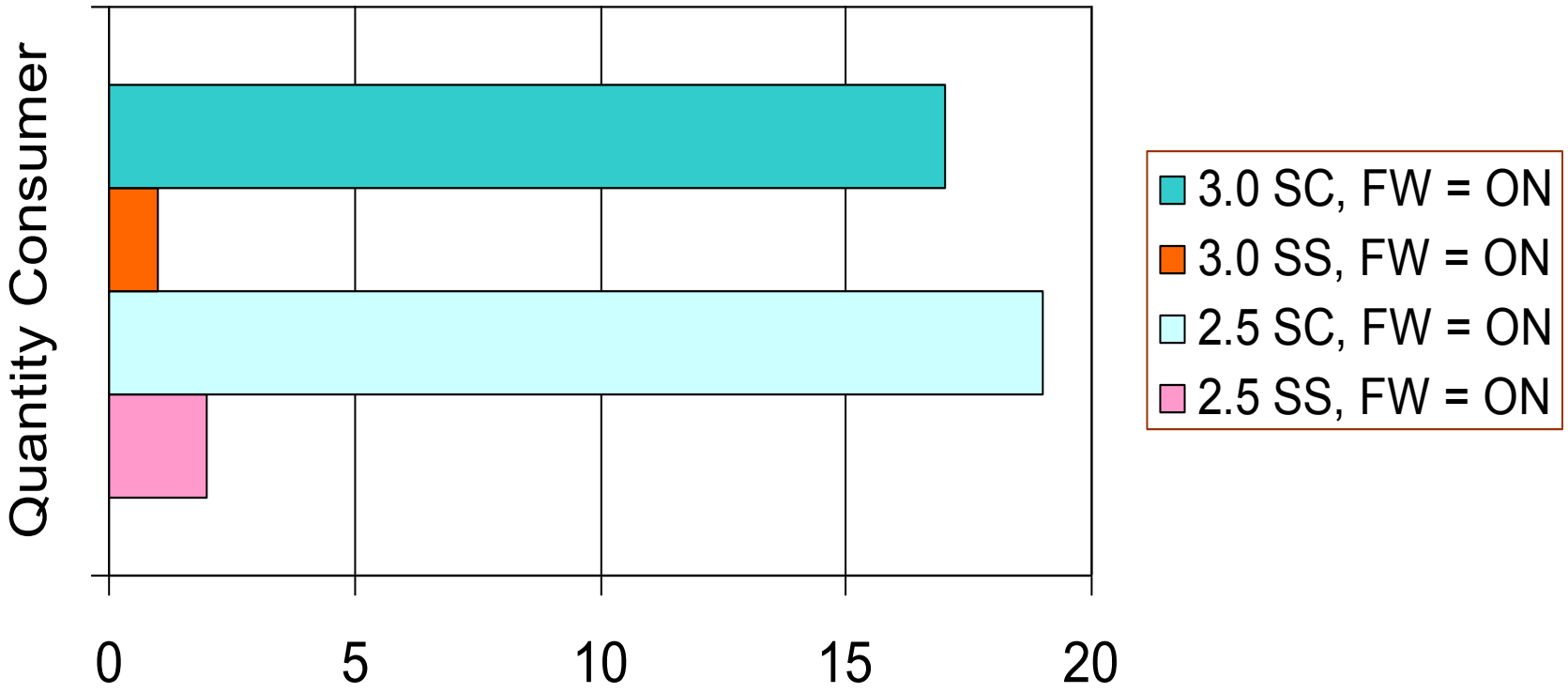


WORKLOAD & RECORD VERSIONS, FW = ON (2/2)

Attaches: 100. Warm-up: 10 min. Measure: 180 min.

Table: **consumer** of quantity after FIFO handling

Total Versions / Total Records, %:

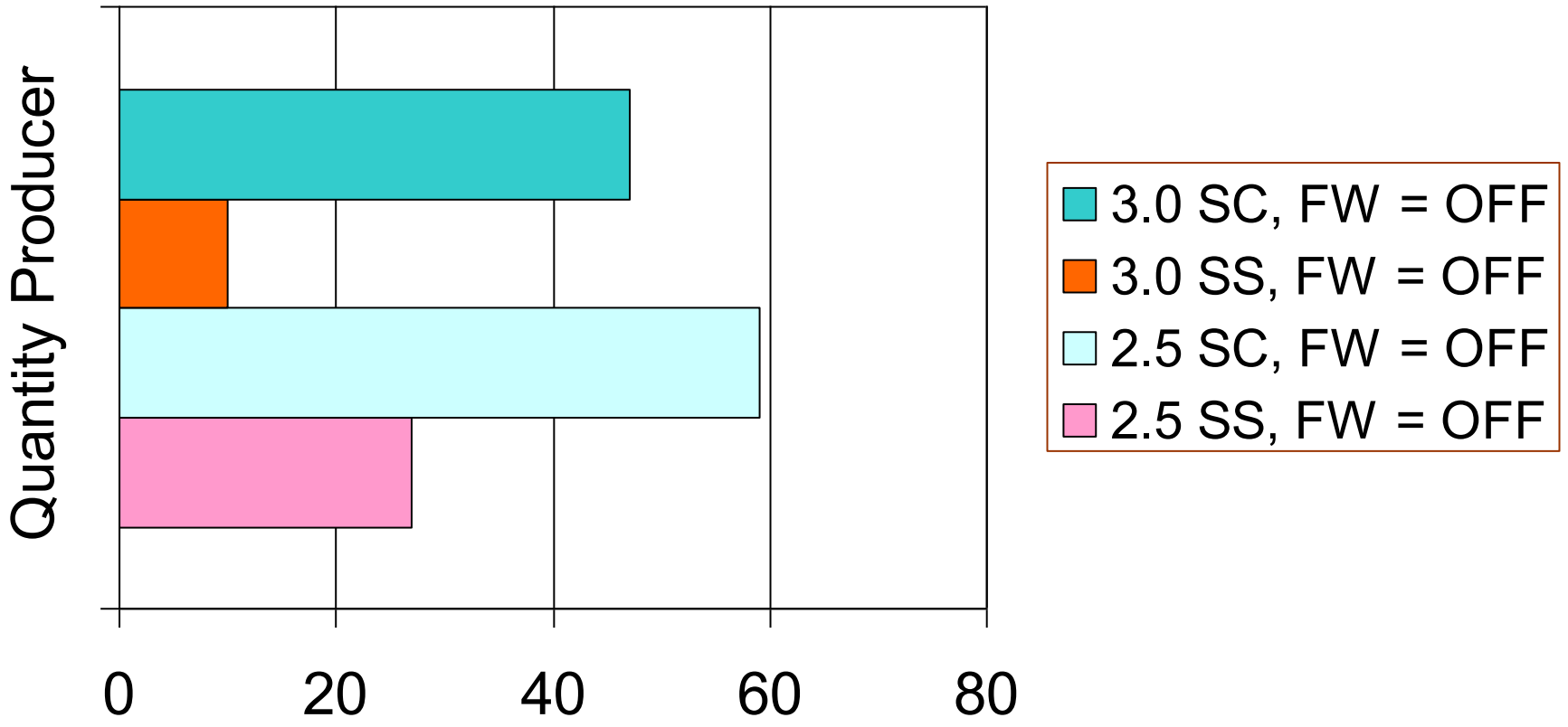


WORKLOAD & RECORD VERSIONS, FW = OFF (1/2)

Attaches: 100. Warm-up: 10 min. Measure: 180 min.

Table: **producer** of quantity for FIFO distribution

Total Versions / Total Records, %:

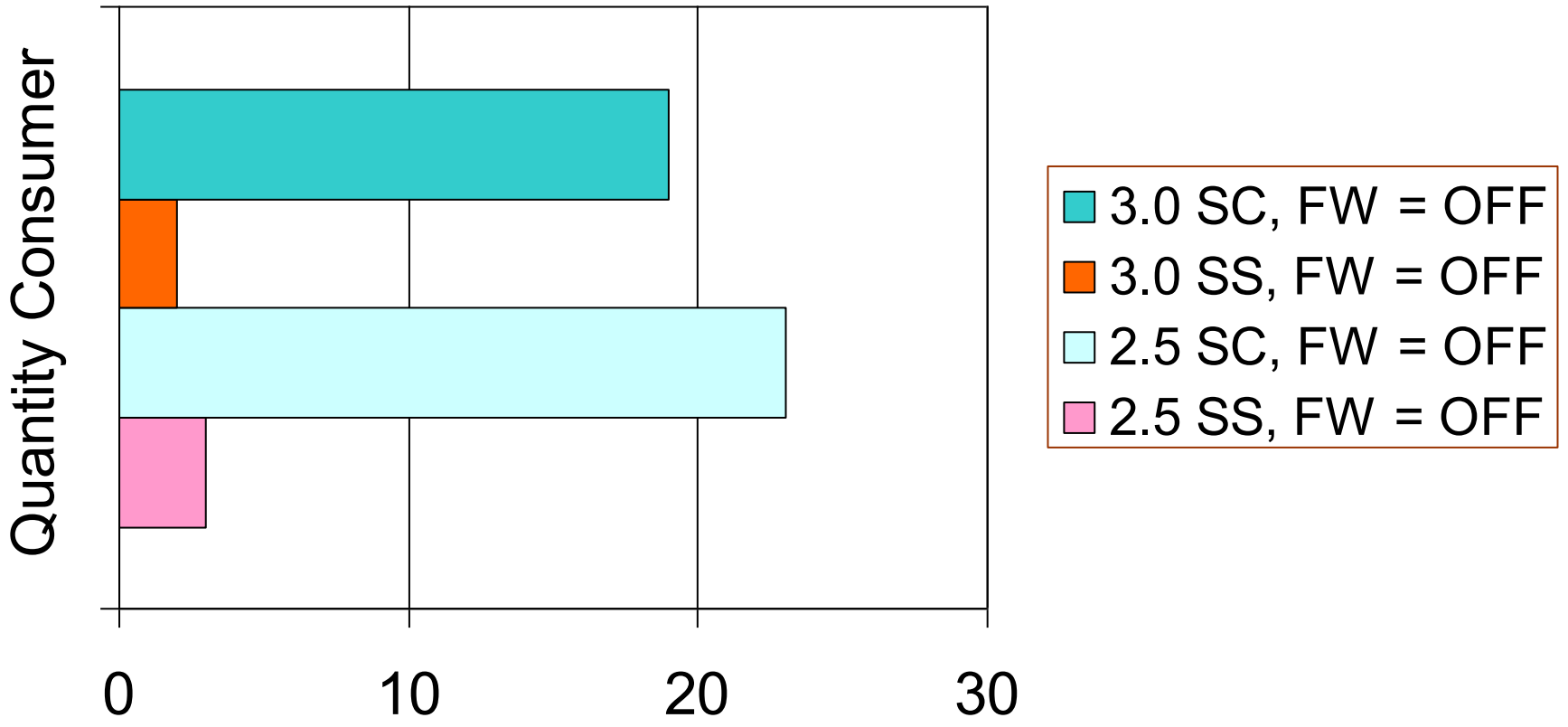


WORKLOAD & RECORD VERSIONS, FW = OFF (2/2)

Attaches: 100. Warm-up: 10 min. Measure: 180 min.

Table: **consumer** of quantity after FIFO handling

Total Versions / Total Records, %:



BUGS & ODDITIES

Total detected bugs: more than 30 (see doc & tracker).

Not fixed yet:

Spontaneous crashes, 3.0 SC, without adding any message in firebird.log. Database appears broken after this. No bugchecks.

Bugchecks with text about ‘wrong record length’, in 3.0 only

“Page type 4 (or 5) lock denied” in firebird.log, in 3.0 only

“I/O error during read file “fb_table_***”, file exists”, in 3.0 only

Standard error messages that should be shipped to client occurs in firebird.log

Attempts of PK violations where bulk of undo occurs (when one of testing machines hangs etc). Firebird 3.0 crashes when workload more than 200 attaches.

GOOD NEWS

- Monitoring was greatly improved in 3.0;
- New monitoring counters and especially table mon$table_stats - the “golden key” in search of performance bottleneck;
- Overall impression about current 3.0: much stable than it was in aug. 2013
- Sounds like paradox but: currently 3.0 SuperServer is more stable than all others (2.5 and 3.0 SuperClassic!)

QUESTIONS?