PGECons
PostgreSQL Enterprise Consortium

Improving PostgreSQL for Greater Enterprise Adoption

Tetsuo Sakata / Nippon Telegraph and Telephone Corporation
Yurie Enomoto / Fujitsu Limited

19.Jun.2015, PGCon in Ottawa

Abstract

- PostgreSQL Enterprise Consortium (PGECons for short) is an organization that consists of major IT companies in Japan, aiming to promote PostgreSQL to enterprise users in the country.
- Since 2012 when PGECons was established, we have been doing surveys of PostgreSQL's functions and performance to PGECons members to estimate how PostgreSQL well meets their requirements.
- In this talk, we will focus on some of major requests from the surveys, including enhancement of table partitioning and error messages handling.
- From the enterprise users' point of view, we would like to share these obstacles behind the requests that might limit PostgreSQL's acceptance, in order to cope with these issues with the community.

Who are we?

- Tetsuo Sakata
  - □ Works for NTT Open Source.
- Yurie Enomoto
  - □ Works for Fujitsu ltd.
- PostgreSQL Enterprise Consortium
  - □ An organization that consists of IT Companies in Japan including Fujitsu, NTT, NEC, SRA OSS and so on.
    - 16 companies for its regular member
  - □ Detailed information;
    - https://www.pgecons.org/en/about/
    - Talk in PGCon 2013
      "Introducing PostgreSQL Enterprise Consortium activities"

Trend of the OSS adoption for enterprise in Japan × Experiences and achievements by more than 5,000 companies

Public Sector

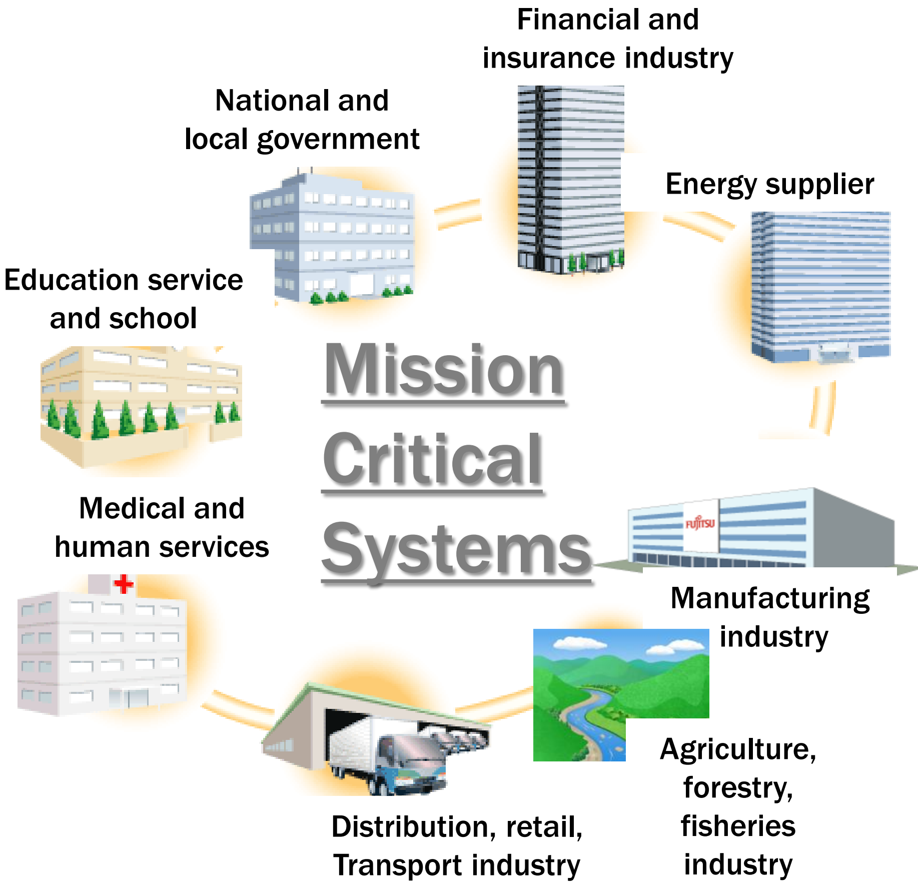
- Prompt using “Open Standard” for the government and municipal system

Energy Supplier

- Prompt adoption of OSS including PostgreSQL for cost reduction

Transportation

- Improve the standard guidelines for PostgreSQL
- Use both OSS and commercial product according to the requirement



Financial and insurance industry

National and local government

Energy supplier

Education service and school

Mission Critical Systems

Medical and human services

Manufacturing industry

Agriculture, forestry, fisheries industry

Distribution, retail, Transport industry

Speed-up for PostgreSQL Enterprise Adoption

PGECons activities for mission critical area

- We surveyed functionality and performance required for mission critical area, and found the following items should be improved.
  - □ partitioning facility:
    The performance of queries on partitioned table is not satisfactory when the number of partition is large enough like several hundreds.
  - □ Facilities for trouble shooting：
    - Enomoto-san will talk.
  - □ Others
    - Will be talked some day.

To accelerate PostgreSQL adoption to Greater Enterpsise

Requirement to partitioning

Case Study for Partitioning usage

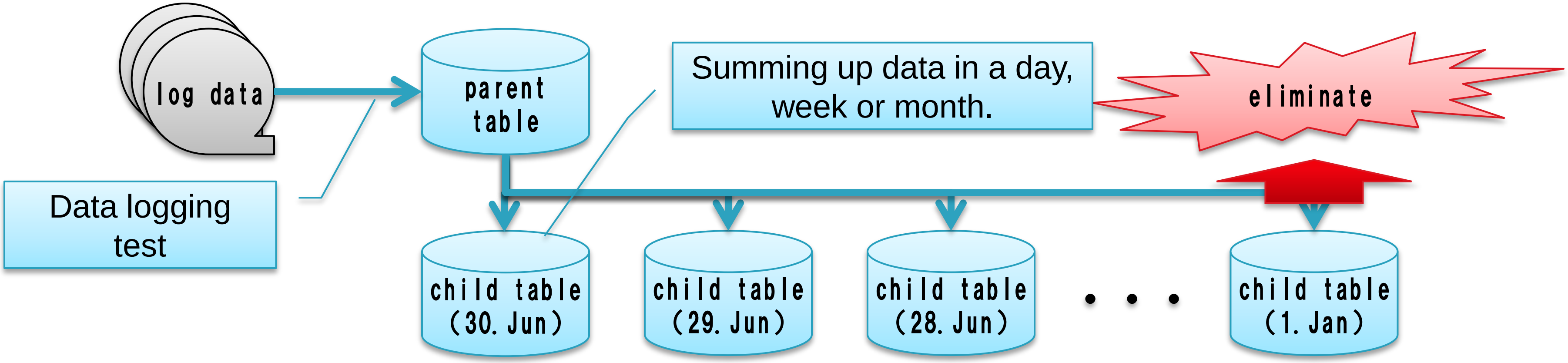
- To clarify partitioning issues, studied its usage
  - □ What are typical usage patterns and cases of partitioning?
  - □ What are required to partitioning features?
- Small survey done by hearing to PGECons members
  - □ Most systems are intended OLTP applications. (OLAP should have different requirement though)
- Two typical application patterns found
  - □ Logging time-series data
  - □ Data divided into branches

Pattern 1: Logging time-series data

- Large DBs that store time-series data like logs
  - Inserted data continuously
  - Aggregate (e.g sum up) them periodically
  - Discard bunch of data periodically
- Cases
  - Telecommunication
    - NTT OSS estimated 1/3 of DBs in the company operated in this way
  - Resale and Financial
    - PGECons members found this kind of operation in resale and financial applications
  - Others
    - An email archiver accelerate searching messages in repository

Logging time-series data with partitioning

- Logged data arrive continuously and are distributed to a partition according to their timestamps.
- Aggregating data in a specified period, we only have to look them up particular partitions not whole the table.
- Discarding data, we only have to truncate partitions.
  - □ We can do it in a short time without vacuuming after.



log data

parent table

Summing up data in a day, week or month.

eliminate

Data logging test

child table (30. Jun)

child table (29. Jun)

child table (28. Jun)

・・・

child table (1. Jan)

Pattern 2: Data divided into branches

- Databases in financial applications store data in partitions. Each partition stores data related to a branch of a bank.
- The number of partitions is about 1000, it depends on number of branches a bank has.
  - Data related to branches can be handed independently.
  - Several queries aggregated branches' data.

Common situations partitioning used

- Database Size
  - □ Databases larger than 1TB require partition facility.
- Typical application
  - □ Logging time-series data and their aggregation
  - □ Data processing is limited to a subset of database (e.g. divided in to branches)
- Queries
  - □ Most queries search only one partition to look up data or aggregate them.
  - □ Several queries aggregate data from more than one partitions or join tuples between partitions.

Main requirement for partitioning

- Number of partitions: 1000 partitions are sufficient to most applications.
  - □ cf. 1 partition per day for 3 years.
  - □ Speedier query execution and easier definition required
- Partition rules: “range” and “list” are used most.
  - □ Some require multi-column partitioning.
  - □ “hash” is seldom used.
- Size of a partition: matches to some dozens of millions tuples.
  - □ From 10GB to 100GB for a partition.

To accelerate PostgreSQL adoption to Greater Enterprise

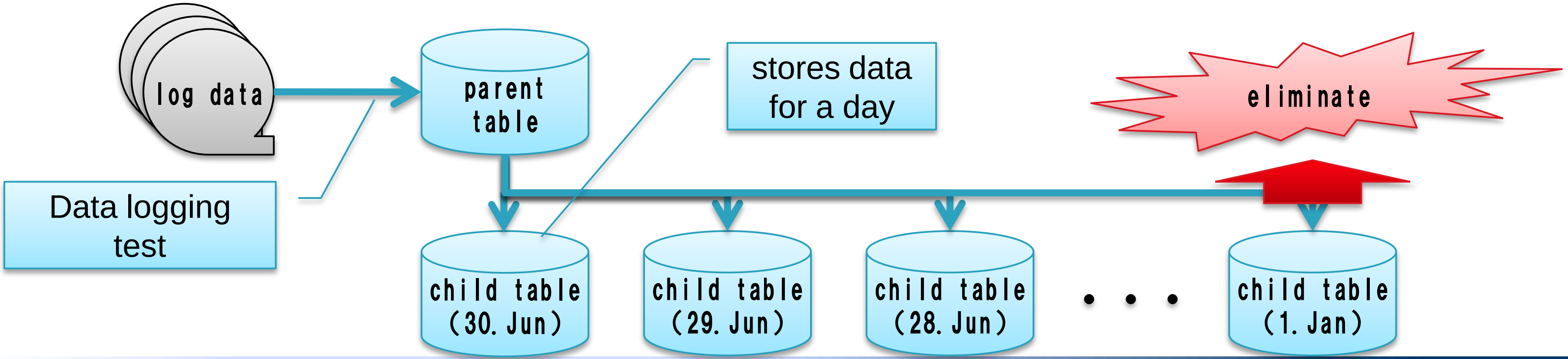
Performance evaluation on partitioning

Performance of partitioning

- PGECons evaluated performance of partitioning using ‘logging time-series data’ pattern to clarify performance issues.
  - □ Various number of partitions were tested to know the slow-down caused by partitioning.
    - Numbers of partitions are 30, 90, 180.
  - □ Various methods to distribute data to partitions are tested to know the overhead of data distribution and the development efforts.
    - the methods are static function, dynamic function and pre-compiled function in C.

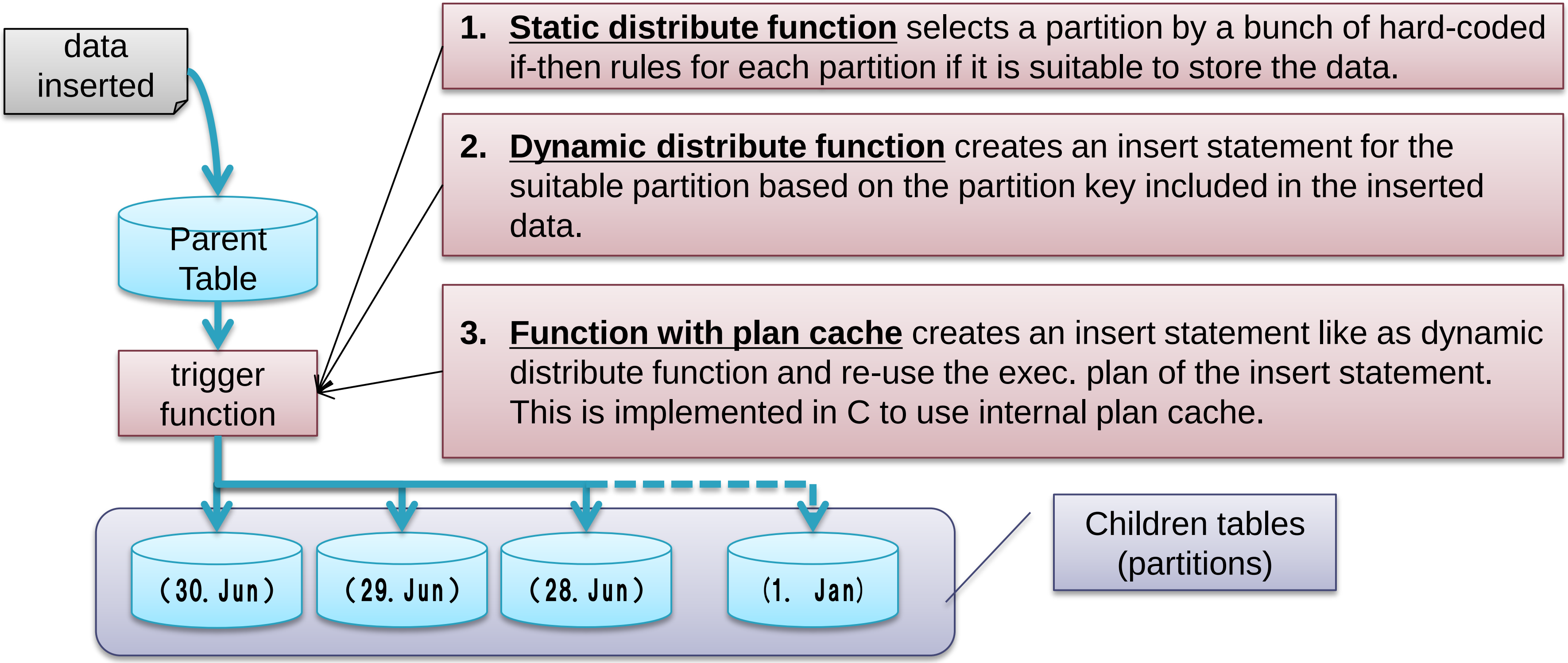
Evaluation Model

- Operation Pattern: ‘logging time-series data’
  - □ Log data from a group of products arrive and are stored into table continuously.
  - □ A partition is prepared for the data arrived in a day.
  - □ data are stored for 1, 3, or 6 months (30, 90 or 180 partitions are used respectively)
- Evaluation Scenarios
  - □ ‘Data logging’ (data insertions to partitions)
    - 3 methods for data distribution are used. They are implemented as trigger functions for performance comparison.
  - □ ‘Data Aggregation’
    - queries placed to aggregate log records arrived in a month (30 days).



Data logging test

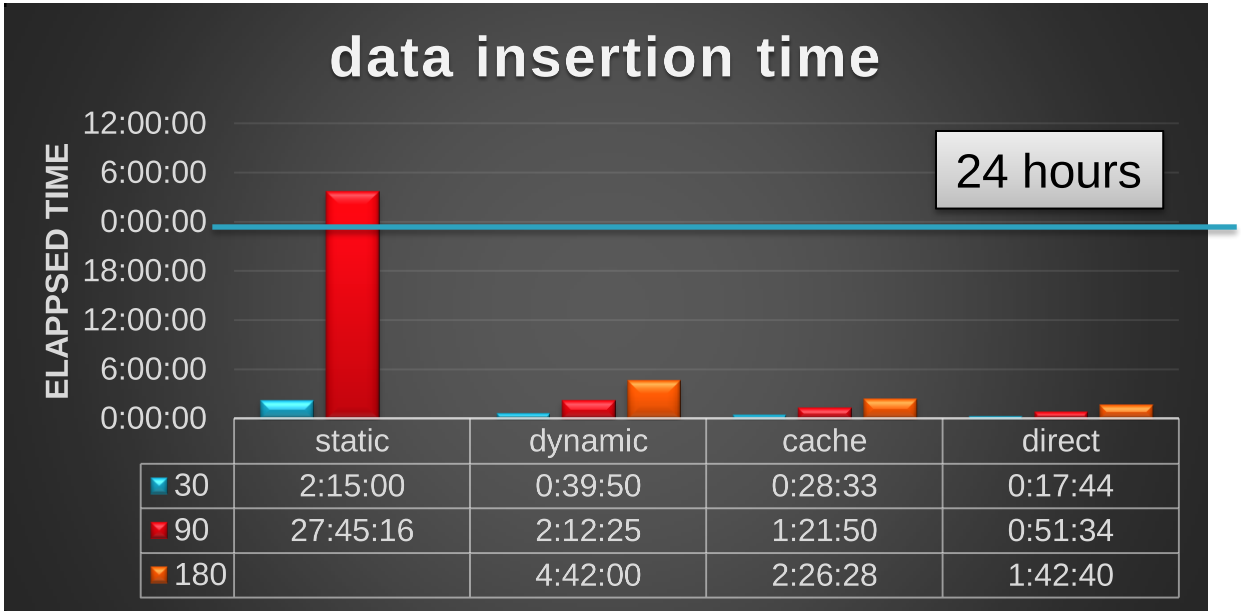
- Amount of inserted data
  - 3 cases are tested; data insertion for 30, 90 and 180 days.
- Data distribution methods;
  - Three methods are used to distribute data to an appropriate partition.
  - And measured insertion time for the same data into a single partition directly for comparison.



1. **Static distribute function** selects a partition by a bunch of hard-coded if-then rules for each partition if it is suitable to store the data.
2. **Dynamic distribute function** creates an insert statement for the suitable partition based on the partition key included in the inserted data.
3. **Function with plan cache** creates an insert statement like as dynamic distribute function and re-use the exec. plan of the insert statement. This is implemented in C to use internal plan cache.

Result of data logging test

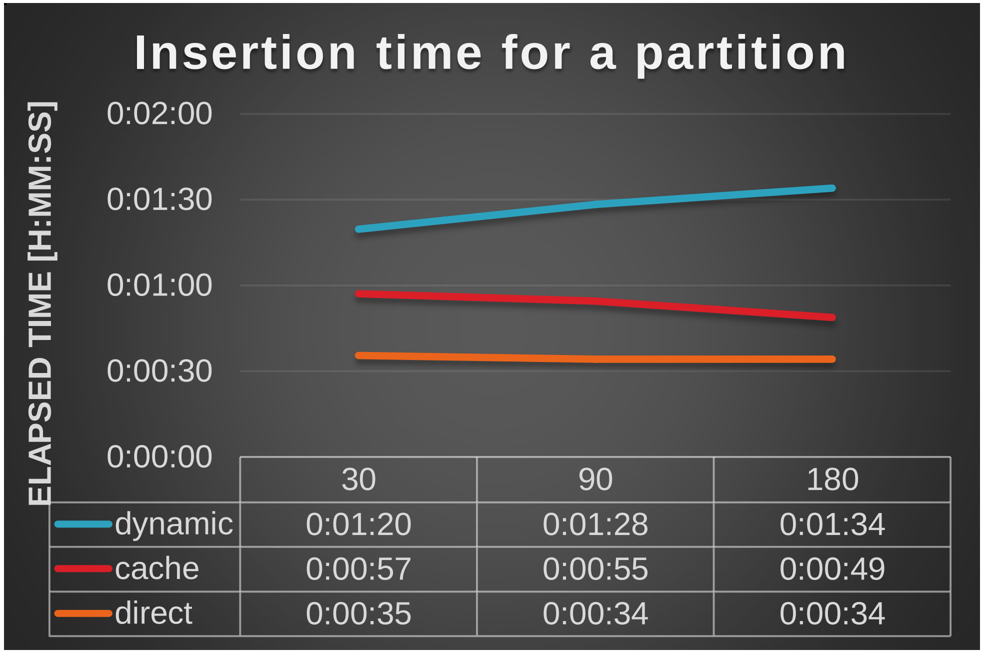
- Elapsed time for data insertion
  Static >> Dynamic > Cache > Direct
  - □ Static function insertion was prohibitively slow. (we gave up testing a case for 180 partition because of limited time)
- Dynamic function performs better than static one and it is easily implemented.
- Function with plan cache outperformed the others, but it required more effort to implement, and a bug would result in DB server’s fault.



	static	dynamic	cache	direct
🟦 30	2:15:00	0:39:50	0:28:33	0:17:44
🟥 90	27:45:16	2:12:25	1:21:50	0:51:34
🟧 180		4:42:00	2:26:28	1:42:40

Result of data logging test (cont‘d)

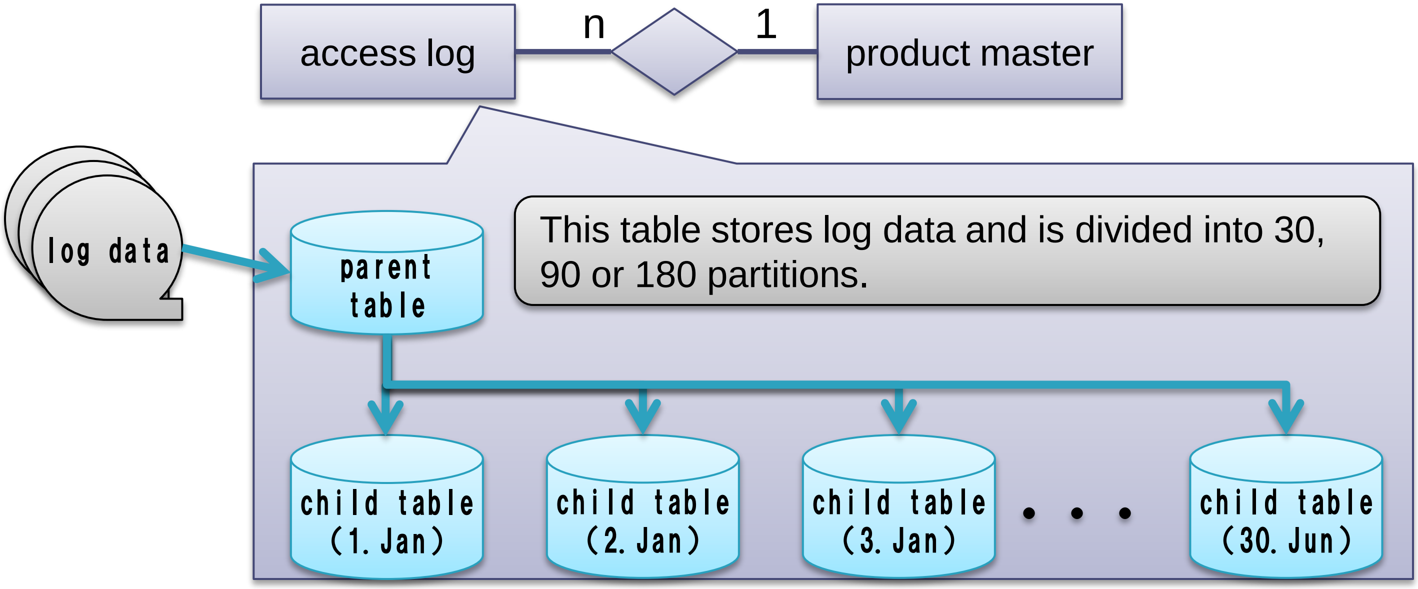
- Comparing 2 methods except static function with regard to insertion time to fulfill a partition.
  - □ These methods insert tuples into partitions in almost constant rate while the number of partitions changes.
  - □ Time for dynamic function grows slowly as number of partitions increase.
- This analysis shows;
  - □ These two methods can be used for a large number of partitions.
  - □ A slight time growth of dynamic distribution would be a latent problem for a large number of partitions



	30	90	180
dynamic	0:01:20	0:01:28	0:01:34
cache	0:00:57	0:00:55	0:00:49
direct	0:00:35	0:00:34	0:00:34

Test for Aggregation

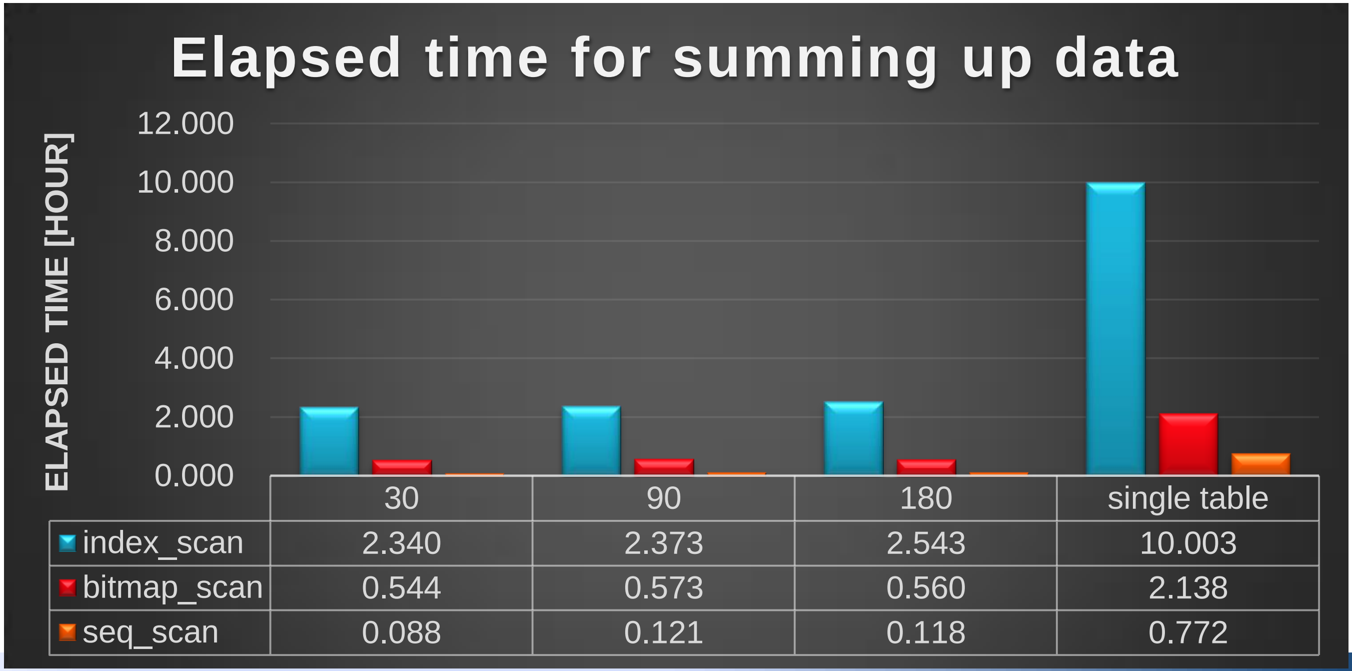
- Query for aggregation
  - □ scans log records arrived in a specified month and aggregate them by product_id and err_code.
  - □ From logged data in 30, 90, 180 days cases, logs arrived in 30 days are aggregated in all cases.
- Tested schema in detail
  - □ Access log record has product_id, date, time, access_time, err_code etc.
  - □ product_master is referred by access_log via product_id.



This table stores log data and is divided into 30, 90 or 180 partitions.

Result of data aggregation

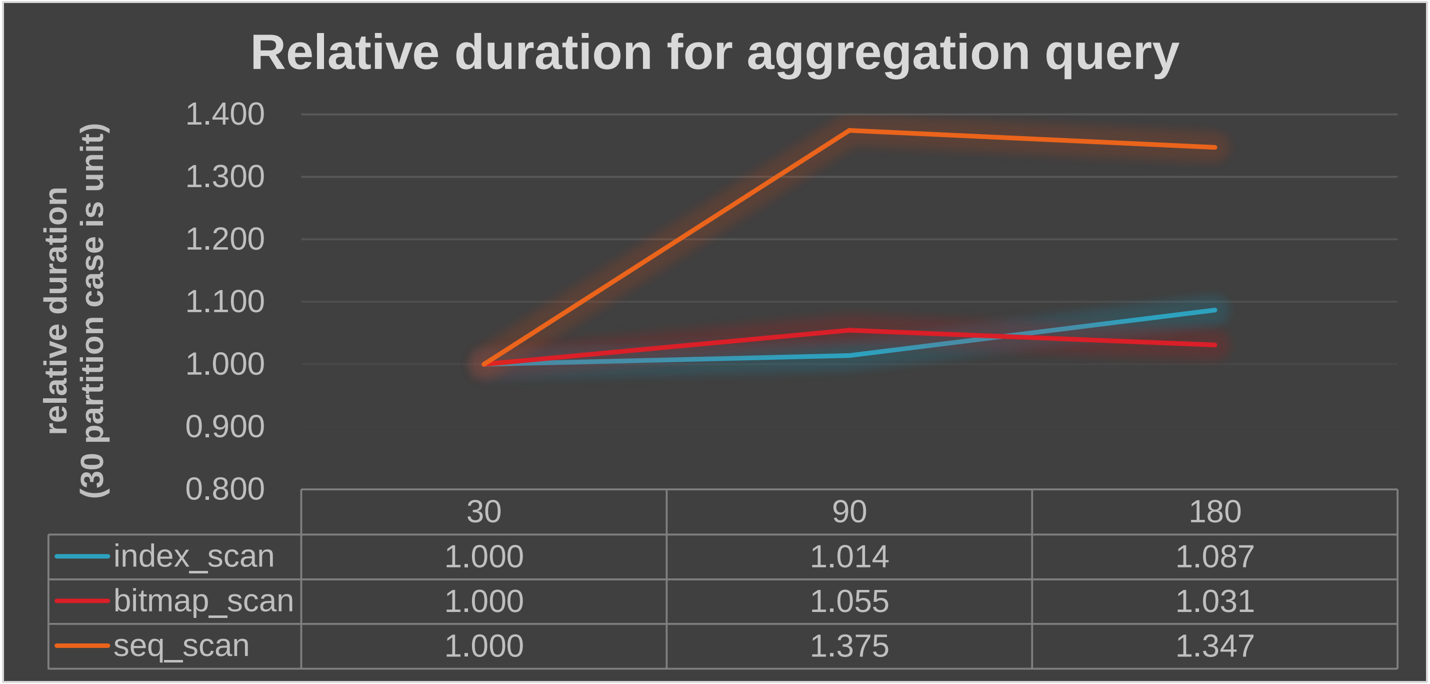
- Queries against 30, 90, 180 partitions that aggregates same amount data.
  - □ case that single table stores all data is tested for comparison (rightmost bars)
- tested three scan methods
  - □ index, bitmap, sequential



	30	90	180	single table
🟦 index_scan	2.340	2.373	2.543	10.003
🟥 bitmap_scan	0.544	0.573	0.560	2.138
🟧 seq_scan	0.088	0.121	0.118	0.772

Result of data aggregation (cont'd)

- Aggregation time growth against num of partitions
  - □ Relative elapsed times with 30 partitions elapsed time being unity.
    - seq_scan slows down obviously
    - Latent problem for large number of partitions used


Relative duration for aggregation query

	30	90	180
index_scan	1.000	1.014	1.087
bitmap_scan	1.000	1.055	1.031
seq_scan	1.000	1.375	1.347

Summary of the evaluation

- Performance evaluation simulates ‘Logging time-series data’ application, which is widely used in enterprise.
- Data insertion can speed up by distribution function with dynamic insert statement creation or using plan cache.
  - With these functions, we estimated about 200 partitions can be used for production system.
- From usability, features for partitioning are not enough to implement application program efficiently.
  - Static data distribution is prohibitively inefficient.
  - Other efficient data distribution methods require more effort to make than static distribution one.

Summary of the evaluation

- Tested aggregation for 180 partitions
  - Aggregate data from the table that has 180 partitions can be done quickly.
- Brief analysis suggested, queries against larger number of partitions will take much longer time.
  - According to hearing, queries against the table that has much more partitions (e.g. 500) can not be executed as quickly as this case.
  - This shows we need improve planning and execution such as constraint exclusion.

Awaiting Solutions

- Partition Definition – Explicit Syntax
  - □ easier definition of partitioning is desirable. e.g. dedicated statement for partition definition.
    - Boundary checks should be automatically.
    - range and list partitioning used popularly, they should be defined easily.
  - □ Shorter notation for a large number of partitions required.
    - Ideally 1000 partitions defined in a few lines, e.g. ...
    - range partition on time-series domain can define partitions for days, months etc. by simple syntax.
    - list partition on a given table of PostgreSQL can define partitions for each row of the table.

Awaiting Solutions

- Effective query execution
  - □ Increase number of partitions up to 1000, it requires more efficient query processing as follows.
- Data distribution
  - □ By now a user defines triggers to distribute data to partitions, the distribution function should be built-in and accelerated
    - Desirable speeding up by binary search and/or indexing
- Query processing
  - □ query for a table has large number of partitions should be executed more effectively.
    - Similar to data distribution, looking up a partition for given value should be accelerated; this should be done easier if we have built-in data structure.

Speed-up for PostgreSQL Enterprise Adoption

- Improving error code -

What needs for Mission Critical System?

Non Stop

- Don’t stop the social infrastructure system.
- Big damage on the social life in the world

Speedy

- In the case of financial,
  half a day stop become the news.
- If the system failure occurs, support people rush to the site within two hours after failure acceptance.

Stable Operation

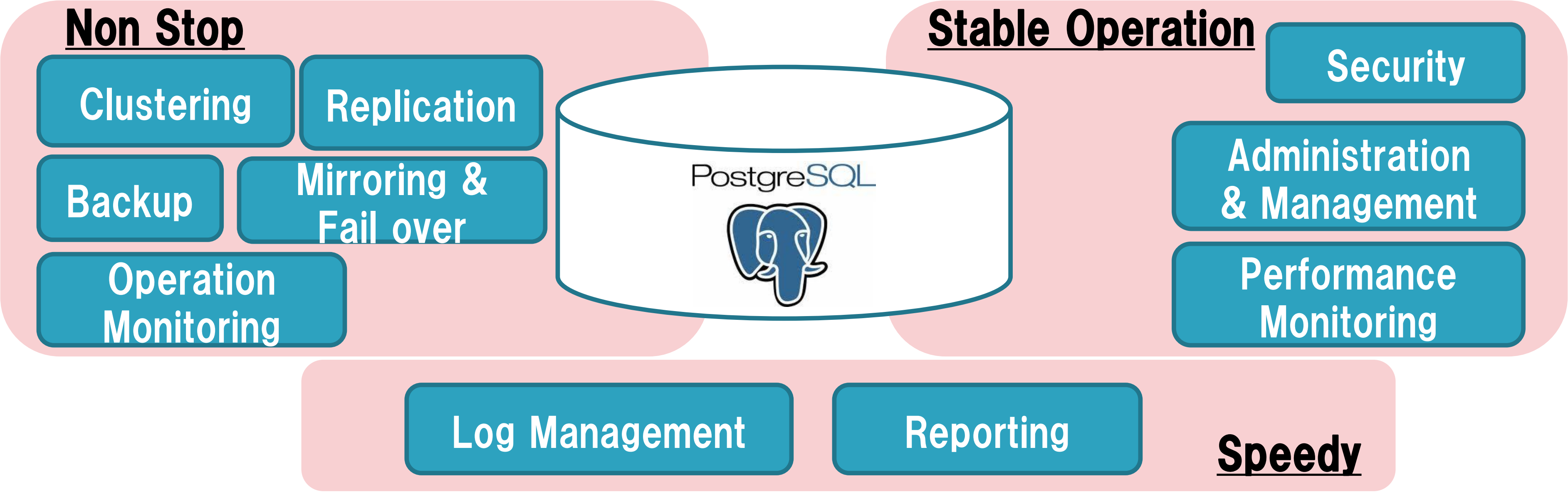
- Stable operation is essential even if the busy period on the season event comes.

24x7 Support

Needs for the construction, operation and support 24 hours a day, every day

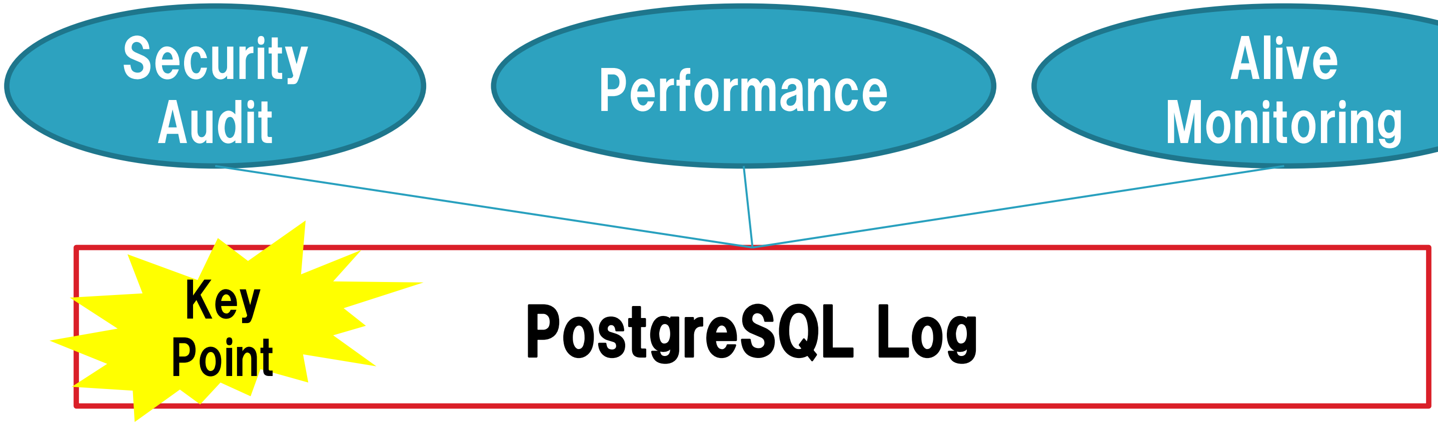
PostgreSQL in Enterprise

- Non Stop : Availability not to stop the system
- Speedy : Responsive identification for cause and solution
- Stable Operation :Always monitor the performance and status of the database, and prevent trouble



It is essential to collaborate with PostgreSQL and software tools.

What information is it important to catch?

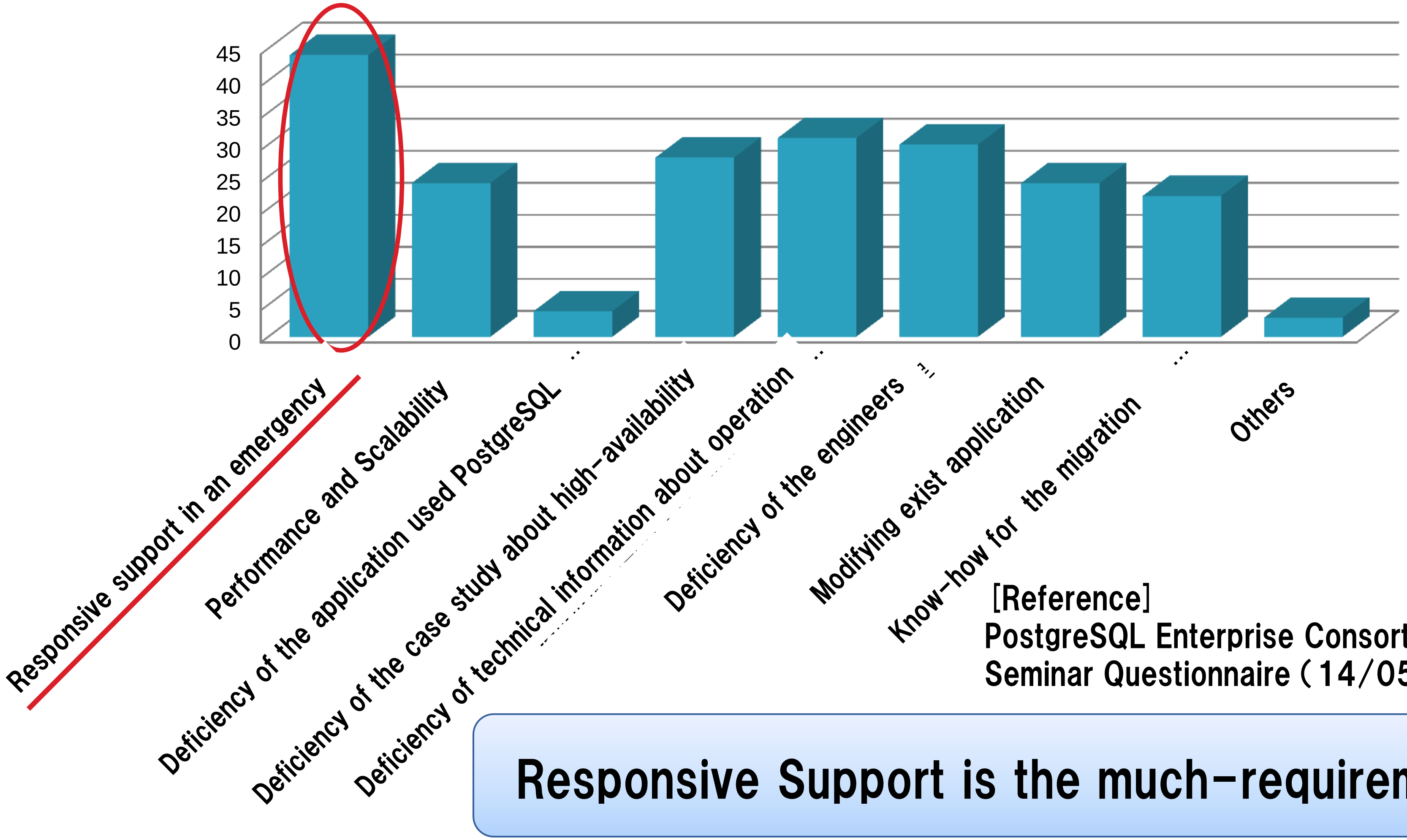


- Significance of PostgreSQL Log

Non Stop	Service alive monitoring, Health check	
Speedy	Quick trouble shooting using error log	Today's Talk Topic
Stable Operation	Performance monitoring (Slow Query) Security Audit	

Much-requirement in Japan

Question. What do you think the difficulty to adopt a more mission-critical area? (multiple answers allowed)



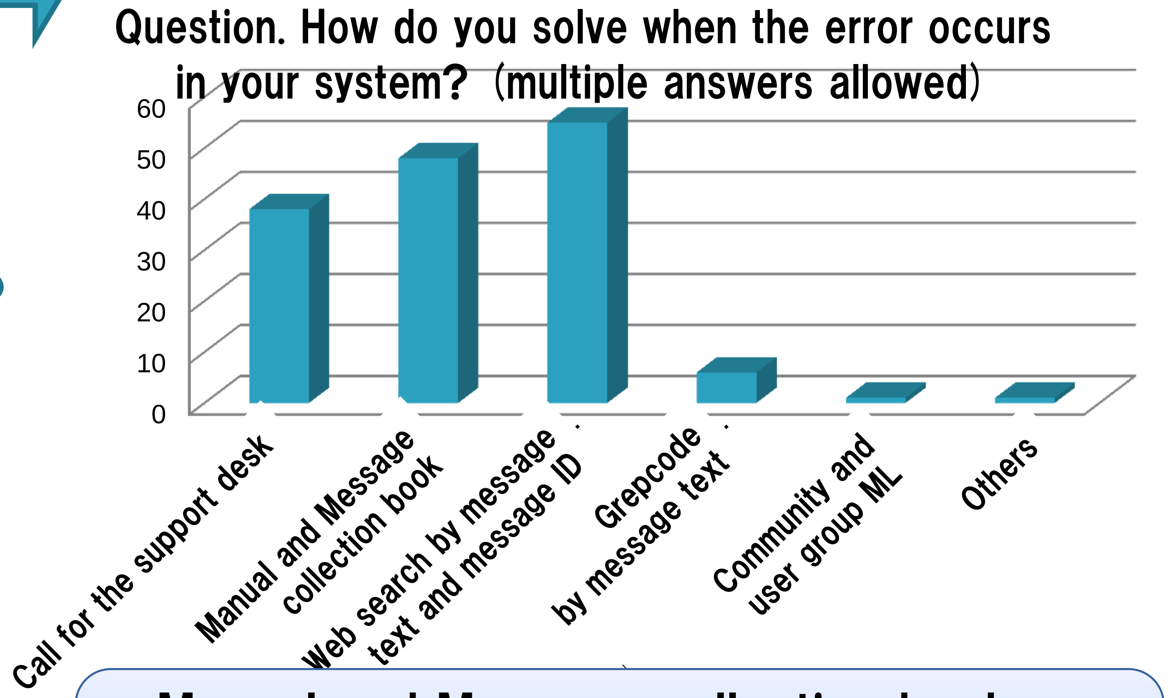
[Reference]
PostgreSQL Enterprise Consortium
Seminar Questionnaire（14/05/2015）

Responsive Support is the much-requirement

How to Troubleshooting?

- If system failure occurs, responsive response is needed.
  - □ Cause Identification
  - □ Action Decision

<u>In the case of using non-PostgreSQL including commercial database</u>


Question. How do you solve when the error occurs in your system? (multiple answers allowed)

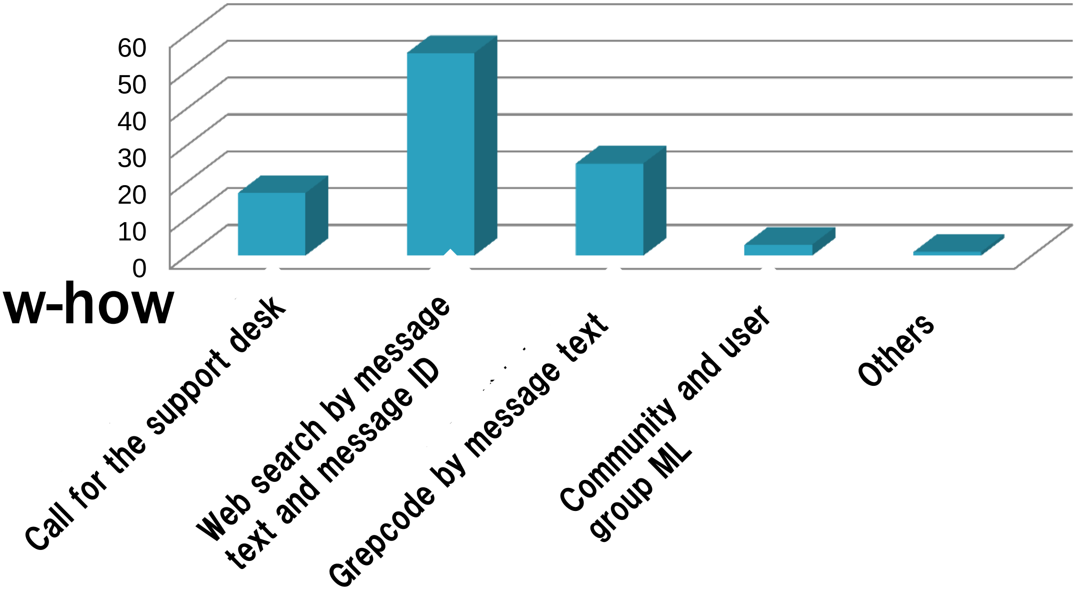
- Manual and Message collection book
- Web search by the error message text or error message ID

PostgreSQL How to Troubleshooting?

- High tendency of investigation by the error message text
  - □ Web research
    - Know-how is here and there.
  - □ GrepCode
    - It takes time to identify the cause.
    - Expert skill and know-how are needed.

In the case of using PostgreSQL

Question. How do you solve when the error occurs in your system? (multiple answers allowed)



Troubleshooting by GrepCode –Case1-

[Using the error code]

- Multi error messages are assigned to one error code.

```
- Example -
Error Code : 40P01 (deadlock_detected)
Pattern1 ”deadlock detected“
        detail : See server log for query details.
Pattern2 ”canceling statement due to conflict with recovery“
        detail : User transaction caused buffer deadlock with recovery.
```

- There are 120 error messages not assigned each error code.

Troubleshooting by GrepCode –Case 2-

[Using the error message text]

- There are 981 message patterns that are output same error message in multiple source code.
 (Max number of duplicated source code : 54)
- When it is included variable number in the message, it is needed to narrow the search by a fuzzy condition and confirm the multiple source code.

Difficulty of PostgreSQL Log

- All log messages are output to one log file.
  It is not be able to the log routing by error level or category.
  - □ Error Trouble
  - □ Slow Query Log

  } Log-scrambling

- The abnormal detection in the error message text
  - □ It is not able to specify the detail conditions and monitor by the pattern matching of a regular expression,
    - We can catch the serious error level of the log
      (PANIC, FATAL, ERROR)
    - It is difficult to catch the significant slow query log.

Ideal Operation of Log Message

① Grant of message ID

- ➢ The error generation part is understood at the component level of the source.
- ➢ Improvement of detection task in operation monitor by the pattern matching

② Uniqueness how to grant of message ID

- ➢ One cause, one message ID
- ➢ Details of SQLSTATE

③ Accumulation of know-how

- ➢ Message ID, Description, System Processing, User's Action
- ➢ Aggregation of troubleshooting know-how sharing by message collection book, website

Classification Example of Conditions based on User's Action

Fujitsu	PostgreSQL (SQLSTATE)
Warning	Class01 (Ex. 01000: warning)
Cancel	57014: query_canceled
Connection Error (protocol violation)	08P01: protocol_violation
Internal Error	XX000: internal_error
Client Configuration Error	F0000: config_file_error
Misunderstanding how to use (Application, command usage)	0A000: feature_not_supported Class42 (Ex. 42804: datatype_mismatch)
Temporary not available	55P03: lock_not_available 40001: serialization_failure
Need Recovery (config file error)	F0000: config_file_error
Need Recovery (Data, Log, Temporary file Error)	XX001: data_corrupted XX002: index_corrupted
Need Recovery (Execute Module Error)	55000: object_not_in_prerequisite_state

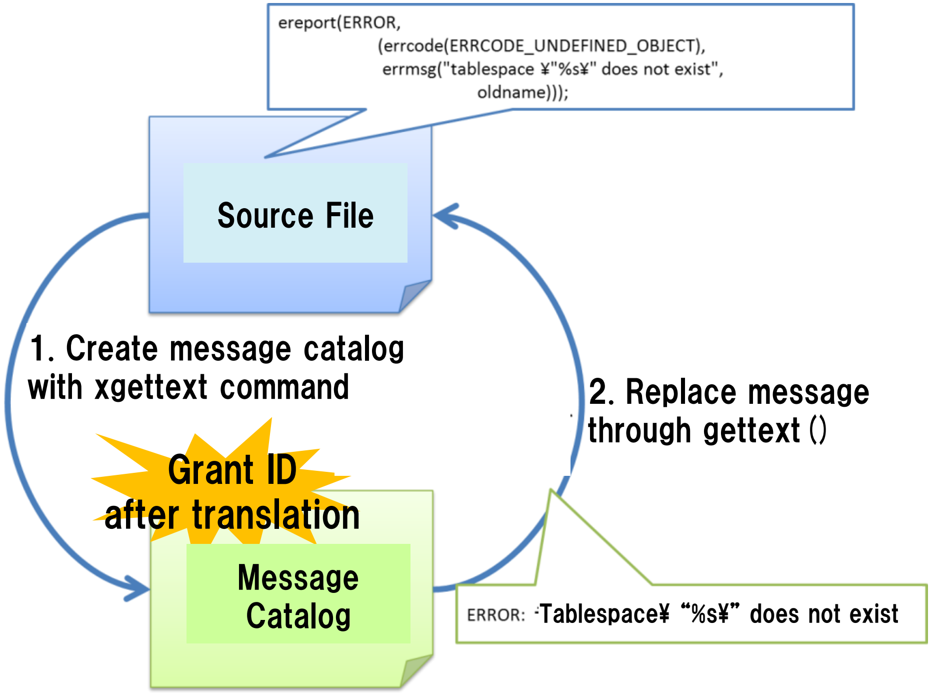
Classification Example of Conditions based on User's Action

Fujitsu	PostgreSQL (SQLSTATE)
Out of resource (Disk)	53100: disk_full
Out of resource (Connection)	53300: too_many_connections
Out of resource (Memory)	53200: out_of_memory
Out of resource (Shared Memory)	53200: out_of_memory
Out of resource (Depth of execution stack)	54001: statement_too_complex
Out of resource (number of file descriptor)	53000: insufficient_resources

[Original additional items]

- Out of resource : Thread, Process, Message Queue, Table lock, File open
- Time out
- Disconnection
- Replication Error

How PostgreSQL Message Catalog works?



```
ereport(ERROR,
              (errcode(ERRCODE_UNDEFINED_OBJECT),
               errmsg("tablespace ¥"%s¥" does not exist",
                             oldname)));
```

Source File
1. Create message catalog with xgettext command
Grant ID after translation
Message Catalog
2. Replace message through gettext ()

ERROR: Tablespace¥ "%s¥" does not exist

- Message Catalog Structure
  - □ Server, JDBC : "PO" File
  - □ .NET : ".resx" File

- Output message mechanism is different between Server and each kind of client driver.
  - □ ereport, elog, ("printf" for stderr)
  - □ JDBC doesn't have error class

[What difficulties?]

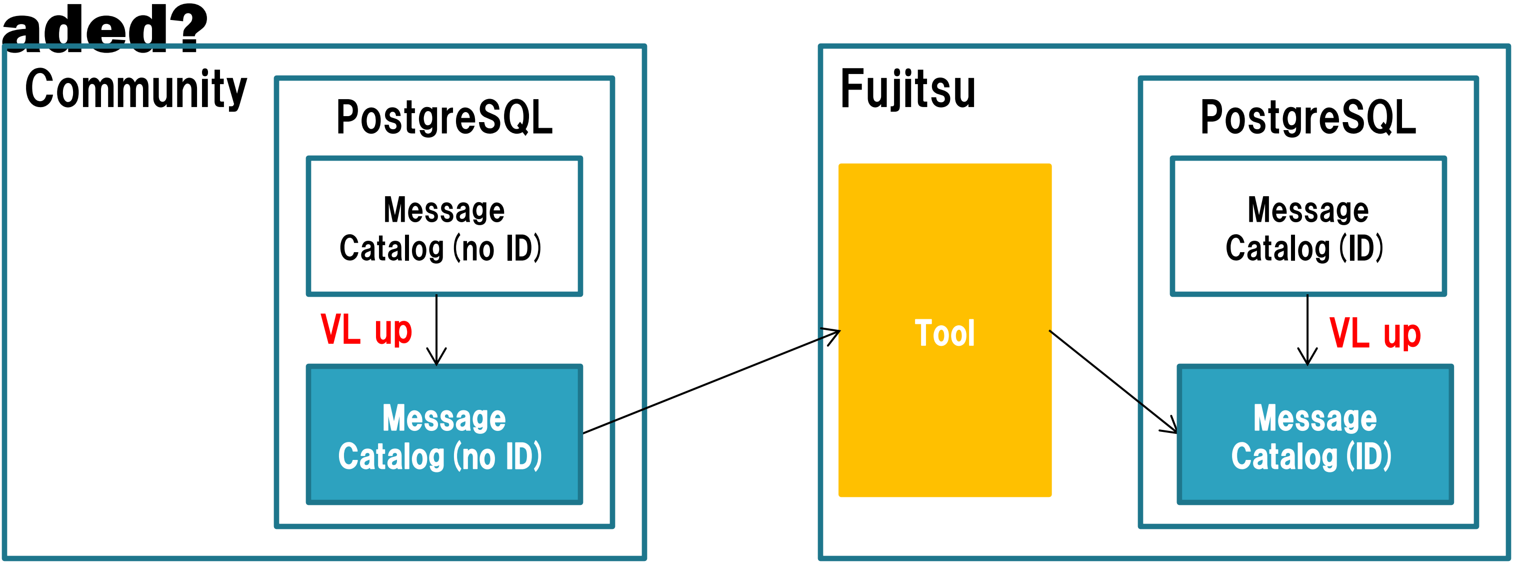
- It is needed to review because there are messages granted wrong error codes.
- Messages that are mismatch between SQLSTATE and kind of error

300 messages

- It is needed to create message catalog separately.
- The messages that are not created catalog are unsupported multiple languages (supported only English).

1,712 messages

How to manage Message ID when PostgreSQL is upgraded?



[Role of tool]

① Extract message differences and calculate mechanically concordance rate based on the vocabulary

② Sort same ID or new ID according to the concordance rate

Because it is matched according to character-string comparison, the result is hardly influenced when message output is changed in source code.

[What Difficulties]

- The incompatibility of message is frequently occurred. (Upgrade from 9.1 to 9.2, there are about 400 incompatibilities about the message of backend)
- There are same messages even though error is different.

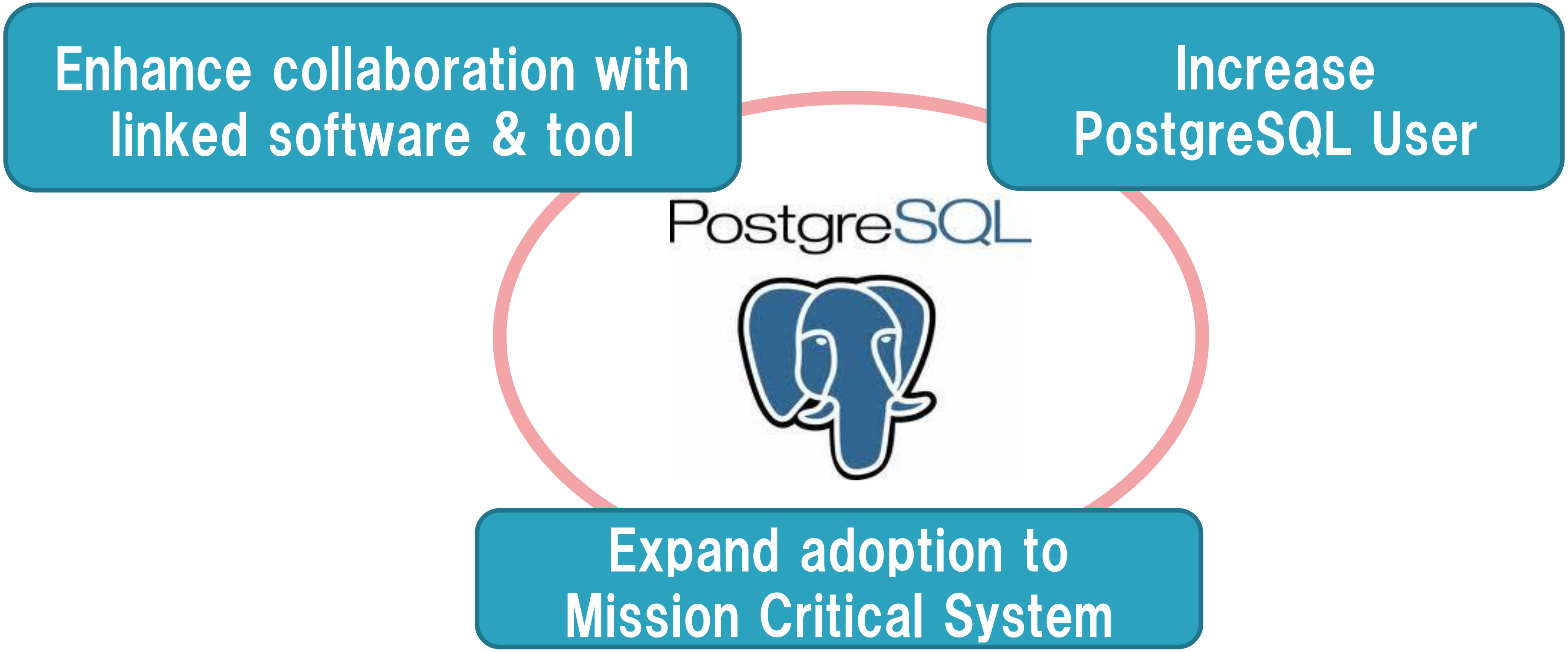
Conclusions

- PostgreSQL log is important role for the stable operation in mission critical system.
  - Operation monitoring, Performance monitoring, Cause Identification
- Effective systematization improvement of log messages in order to strengthen cooperation with peripheral software tools

[Results]

- Categorize PostgreSQL log into the error log and slow query log by message ID（Normally, detection of error log type by string matching)
- Speed-up to identify the cause of error by message ID and ID collection book
- Improve user self-solution for the consolidation know-how by sharing the response collection book

Further PostgreSQL Evolution



Enhance collaboration with linked software & tool

Increase PostgreSQL User

Expand adoption to Mission Critical System

PGECons
PostgreSQL Enterprise Consortium

Thank you