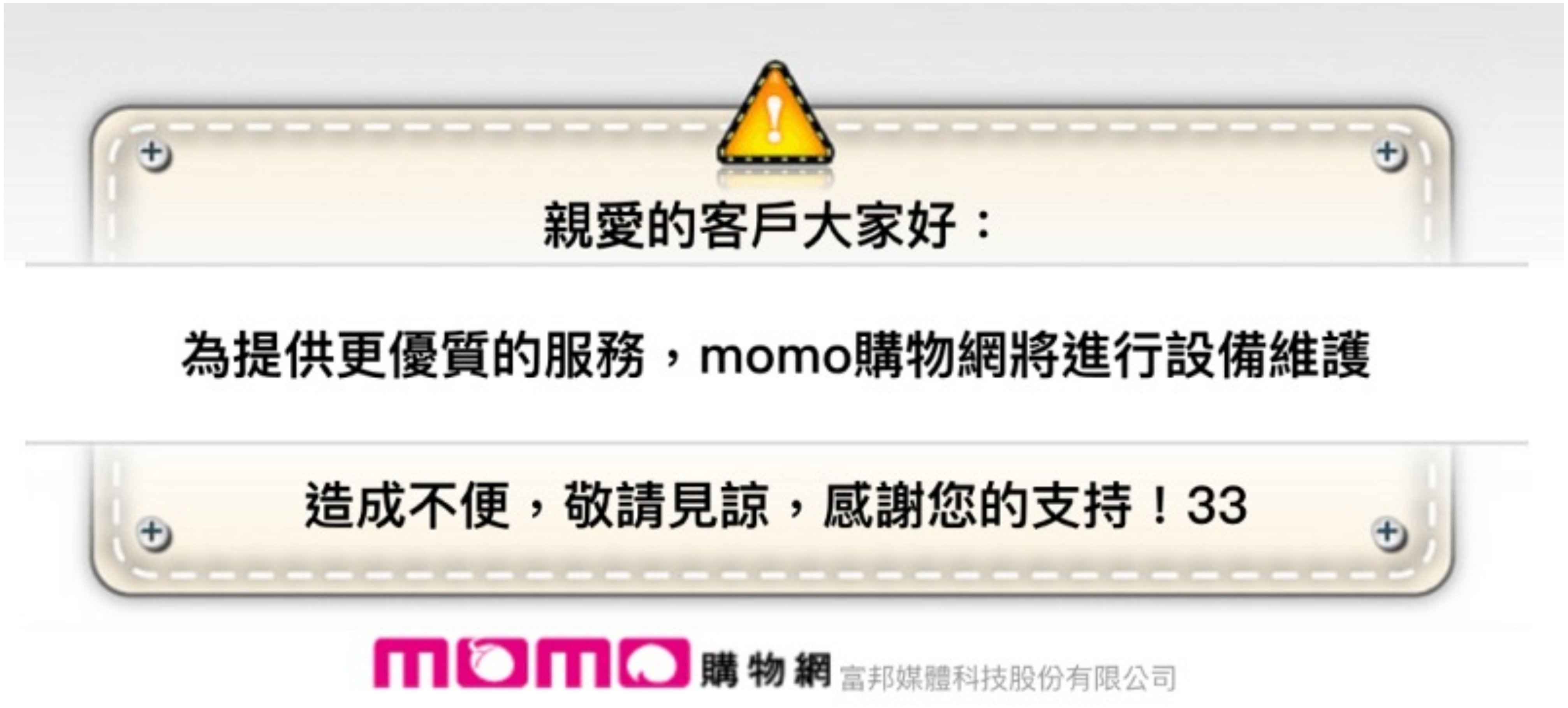
Physical Database Design & Performance Tuning

政治大學
資訊科學系
沈錳坤



親愛的客戶大家好：

為提供更優質的服務，momo購物網將進行設備維護

造成不便，敬請見諒，感謝您的支持！33

momo 購物網 富邦媒體科技股份有限公司

今天是雙11，不過台灣兩大電商平台PChome跟momo卻在今天疑似乘載不了突然湧入的壓力而掛點，我想平台與技術團隊應該壓力山大，但我覺得這是一件很好的事，所以要特別恭喜他們。

gipi的商業思維筆記
學習/閱讀/思考/傳道

Follow

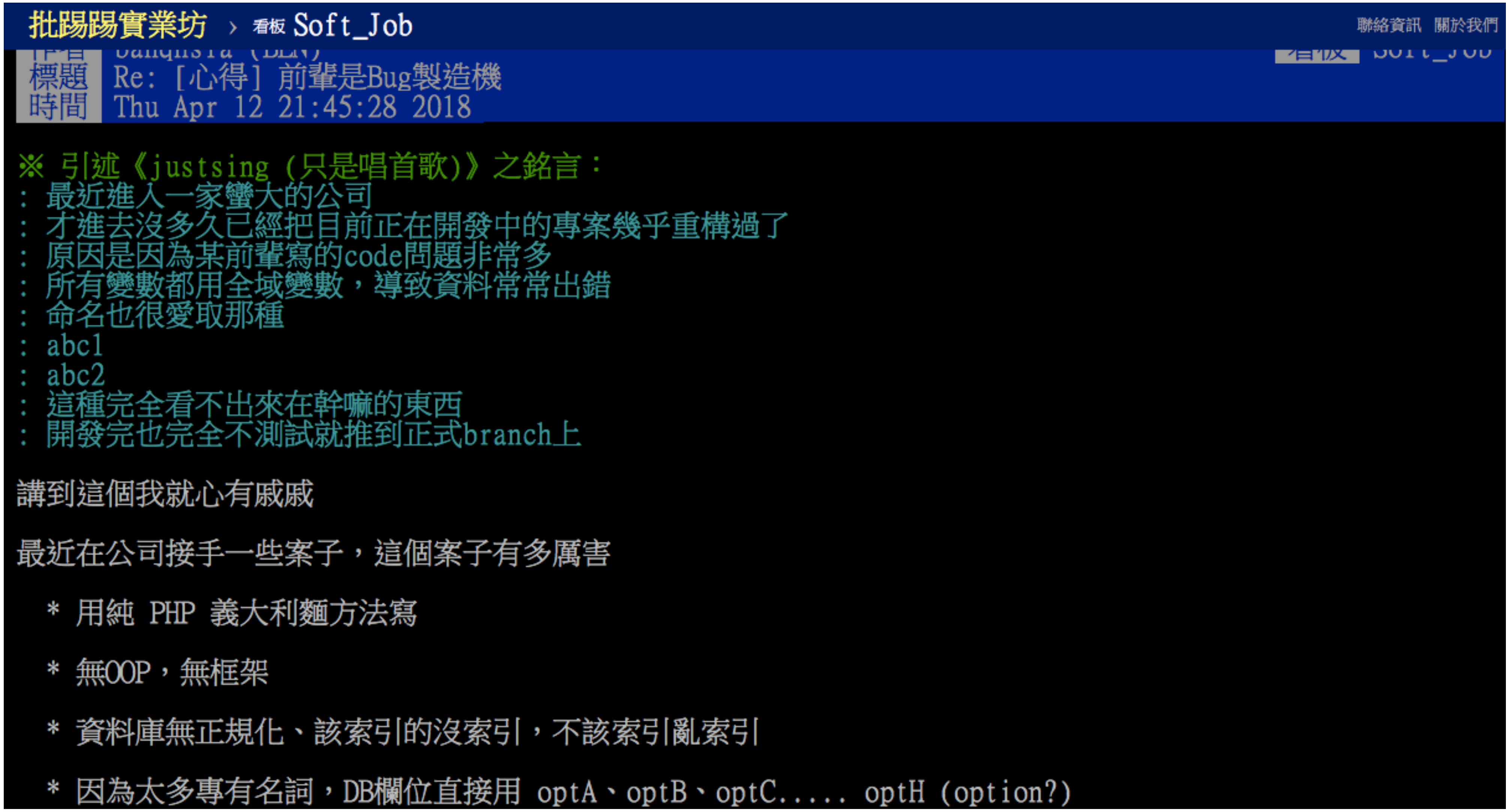
> 不是足夠大的公司，不具有這樣的規模量，你還沒機會遭遇此問題，就如我們過去說的，**技術債是屬於那些活下來的公司，至於那些撐不下去的，技術債跟你一點關係也沒有。**換句話說，就是你夠大，你才有機會碰到這樣的問題。

900

大多數一流的網路公司都曾發生過大規模的系統問題，差別只在於局部崩潰或是全面性的崩潰，但在它們長到這麼大之前，這種異常問題還會少見嗎？一點也不，AWS、Facebook、阿里雲、Netflix、LinkedIn這些公司其實都發生過大規模的異常事件，這些公司的工程師的高水平我想大家都略知一二。

> 一家公司的技術水平，往往都是在遭遇到營運面的困難時，才踏上加速突破的道路。

台灣網路圈這些年來因為市場規模的關係，網站的交易量(transaction)與併發用戶量(concurrent user)一直都不會太大，每秒上千個交易或10萬個併發用戶已經是一線網站，基本上很少有機會讓大家實戰演練一下如何搞定每秒十萬交易跟千萬級併發用戶。



標題 Re: [心得] 前輩是Bug製造機
時間 Thu Apr 12 21:45:28 2018

※ 引述《justsing (只是唱首歌)》之銘言：
: 最近進入一家蠻大的公司
: 才進去沒多久已經把目前正在開發中的專案幾乎重構過了
: 原因是因為某前輩寫的code問題非常多
: 所有變數都用全域變數，導致資料常常出錯
: 命名也很愛取那種
: abc1
: abc2
: 這種完全看不出來在幹嘛的東西
: 開發完也完全不測試就推到正式branch上

講到這個我就心有戚戚

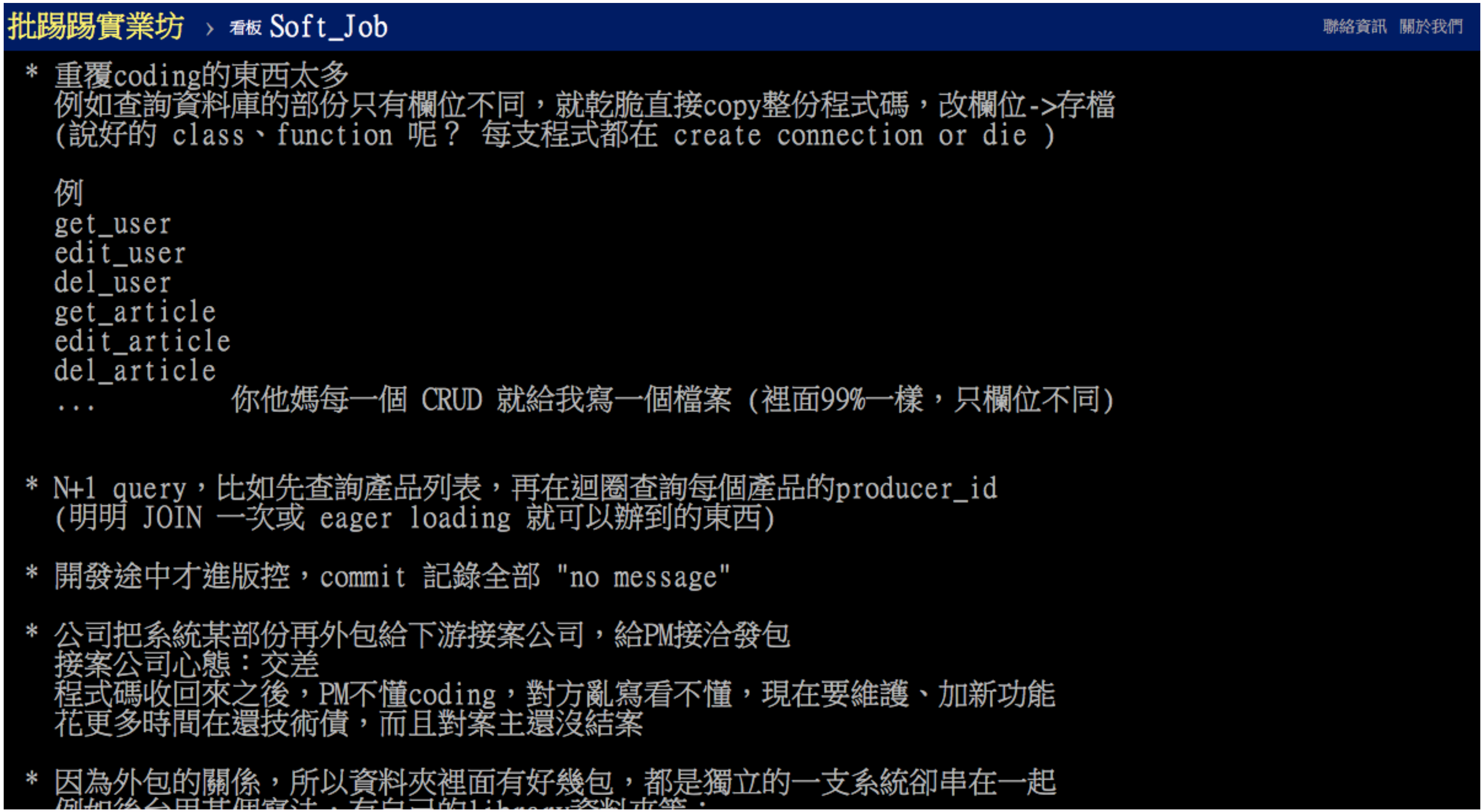
最近在公司接手一些案子，這個案子有多厲害

* 用純 PHP 義大利麵方法寫

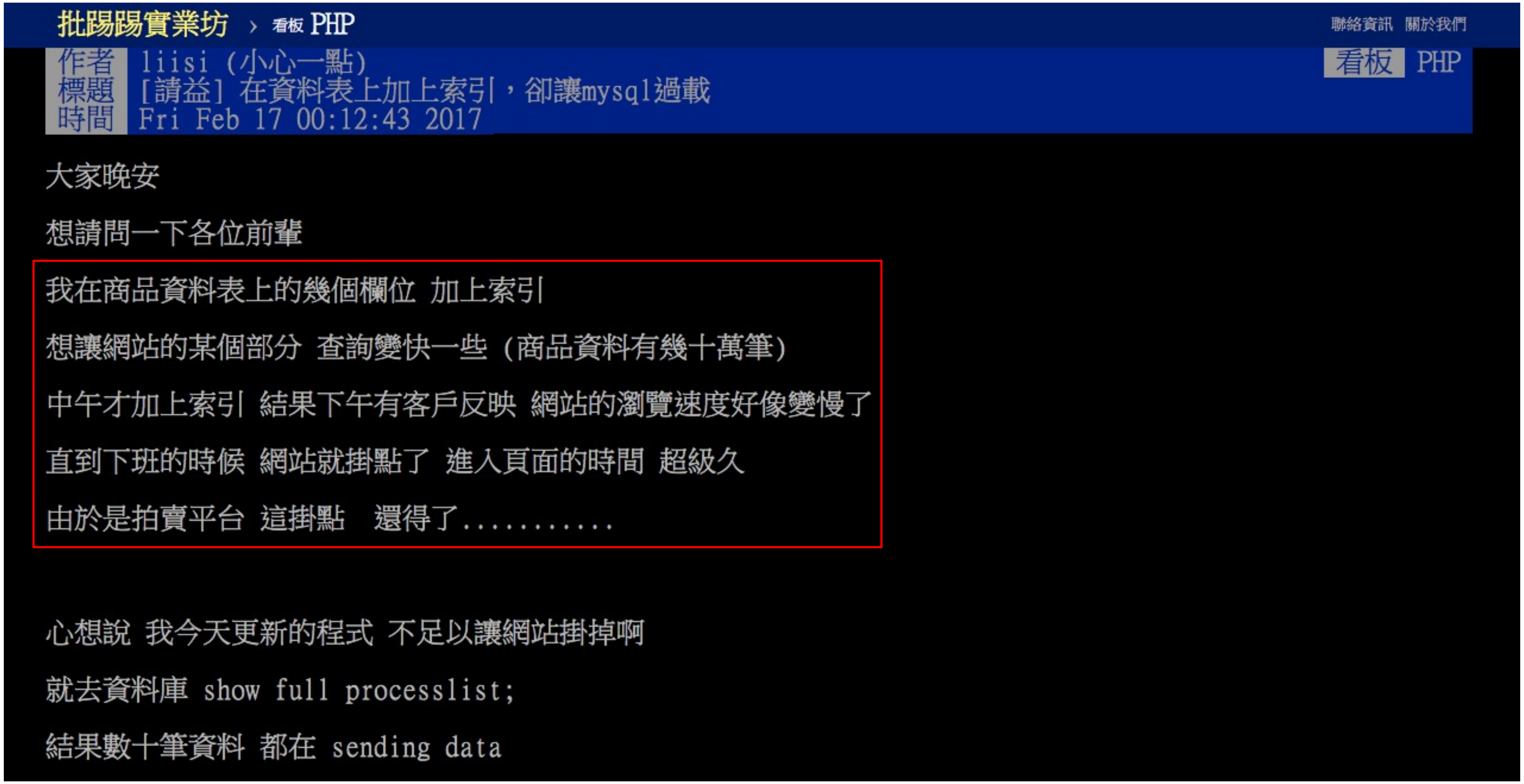
* 無OOP，無框架

* 資料庫無正規化、該索引的沒索引，不該索引亂索引

* 因為太多專有名詞，DB欄位直接用 optA、optB、optC..... optH (option?)



```
* 重覆coding的東西太多
  例如查詢資料庫的部份只有欄位不同，就乾脆直接copy整份程式碼，改欄位->存檔
  (說好的 class、function 呢？ 每支程式都在 create connection or die )

  例
  get_user
  edit_user
  del_user
  get_article
  edit_article
  del_article
  ...         你他媽每一個 CRUD 就給我寫一個檔案 (裡面99%一樣，只欄位不同)


* N+1 query，比如先查詢產品列表，再在迴圈查詢每個產品的producer_id
  (明明 JOIN 一次或 eager loading 就可以辦到的東西)

* 開發途中才進版控，commit 記錄全部 "no message"

* 公司把系統某部份再外包給下游接案公司，給PM接洽發包
  接案公司心態：交差
  程式碼收回來之後，PM不懂coding，對方亂寫看不懂，現在要維護、加新功能
  花更多時間在還技術債，而且對案主還沒結案

* 因為外包的關係，所以資料夾裡面有好幾包，都是獨立的一支系統卻串在一起
```


批踢踢實業坊 › 看板 PHP 聯絡資訊 關於我們

作者 liisi (小心一點) 看板 PHP
標題 [請益] 在資料表上加上索引，卻讓mysql過載
時間 Fri Feb 17 00:12:43 2017

大家晚安

想請問一下各位前輩

我在商品資料表上的幾個欄位 加上索引

想讓網站的某個部分 查詢變快一些 (商品資料有幾十萬筆)

中午才加上索引 結果下午有客戶反映 網站的瀏覽速度好像變慢了

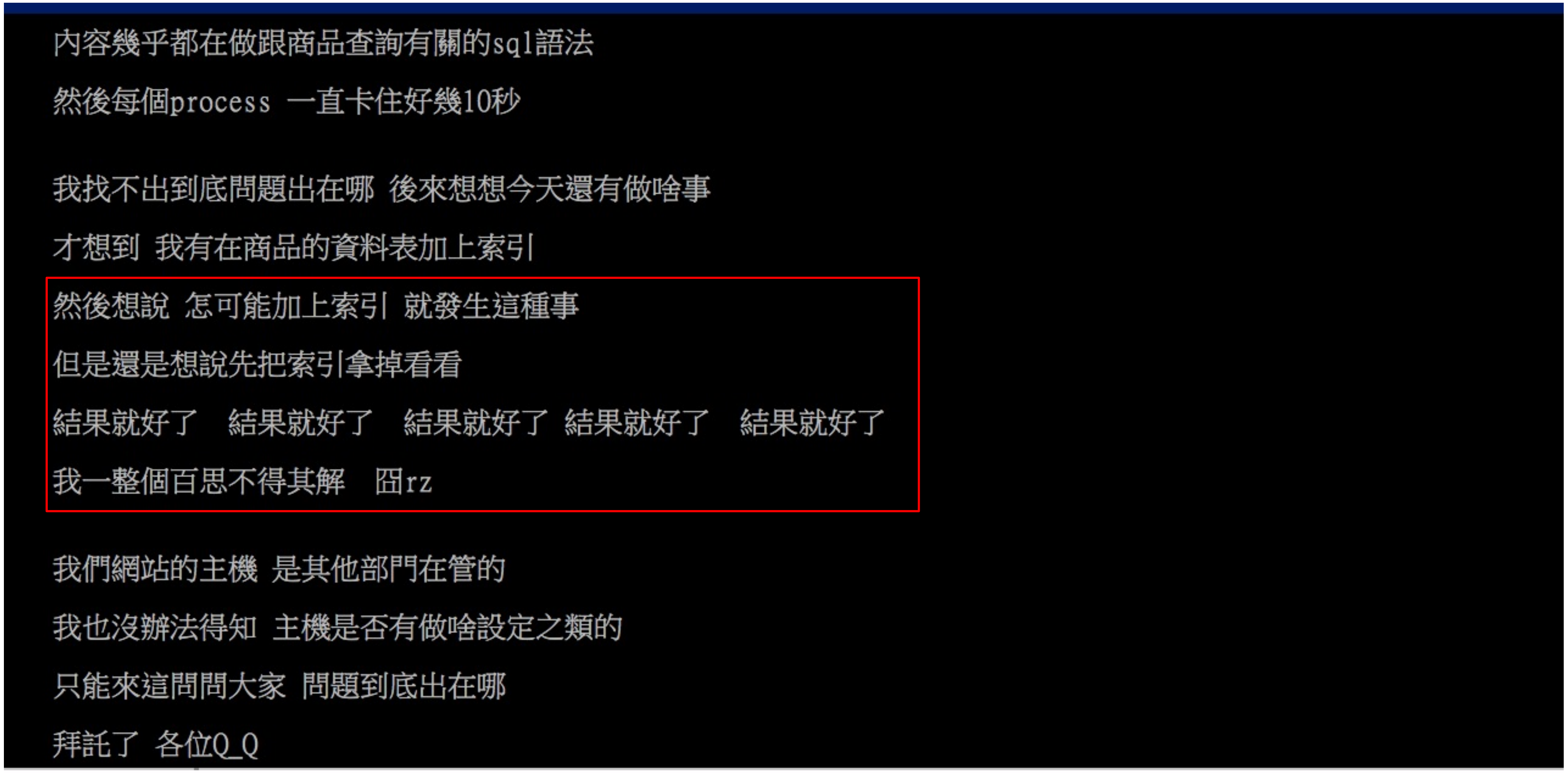
直到下班的時候 網站就掛點了 進入頁面的時間 超級久

由於是拍賣平台 這掛點 還得了...........

心想說 我今天更新的程式 不足以讓網站掛掉啊

就去資料庫 show full processlist;

結果數十筆資料 都在 sending data


內容幾乎都在做跟商品查詢有關的sql語法

然後每個process 一直卡住好幾10秒

我找不出到底問題出在哪 後來想想今天還有做啥事

才想到 我有在商品的資料表加上索引

然後想說 怎可能加上索引 就發生這種事

但是還是想說先把索引拿掉看看

結果就好了 結果就好了 結果就好了 結果就好了 結果就好了

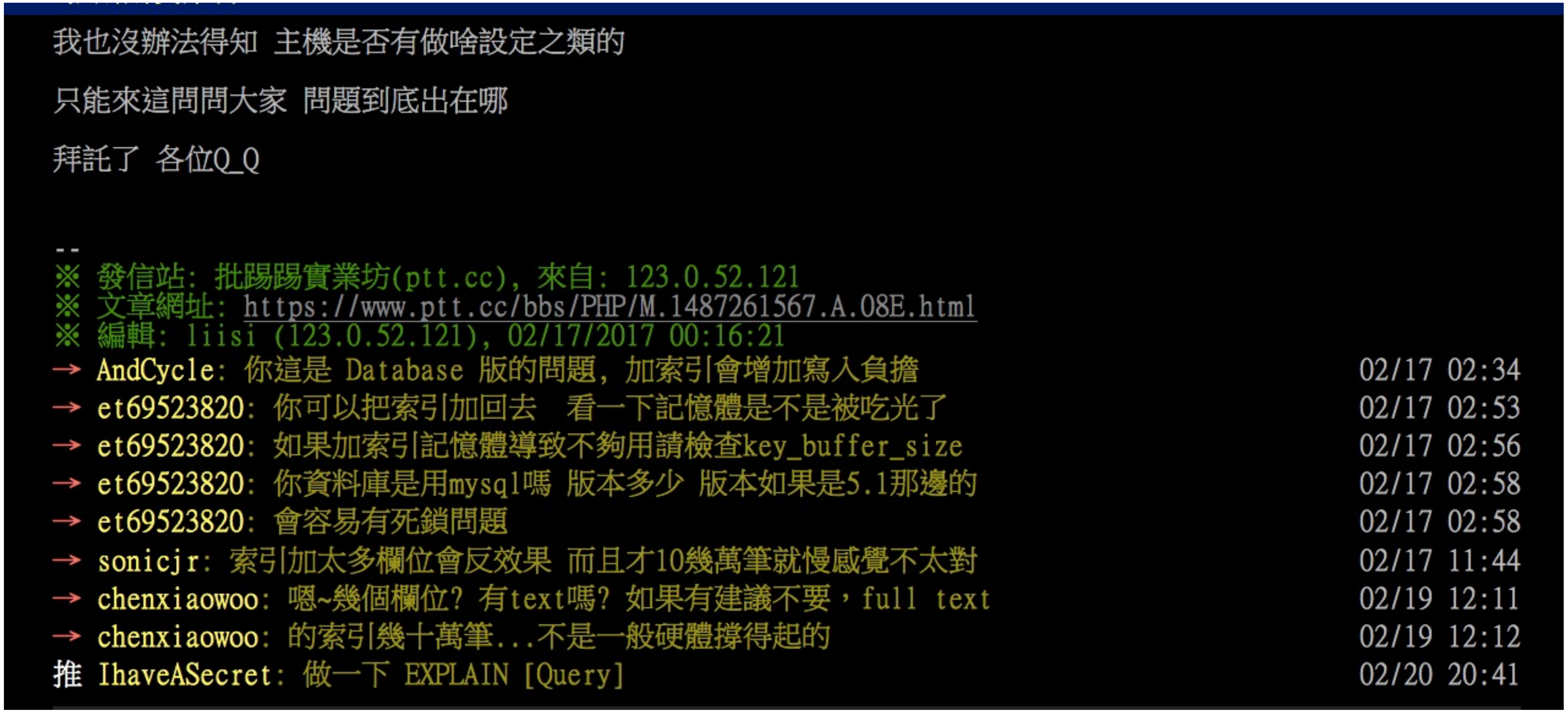
我一整個百思不得其解 囧rz

我們網站的主機 是其他部門在管的

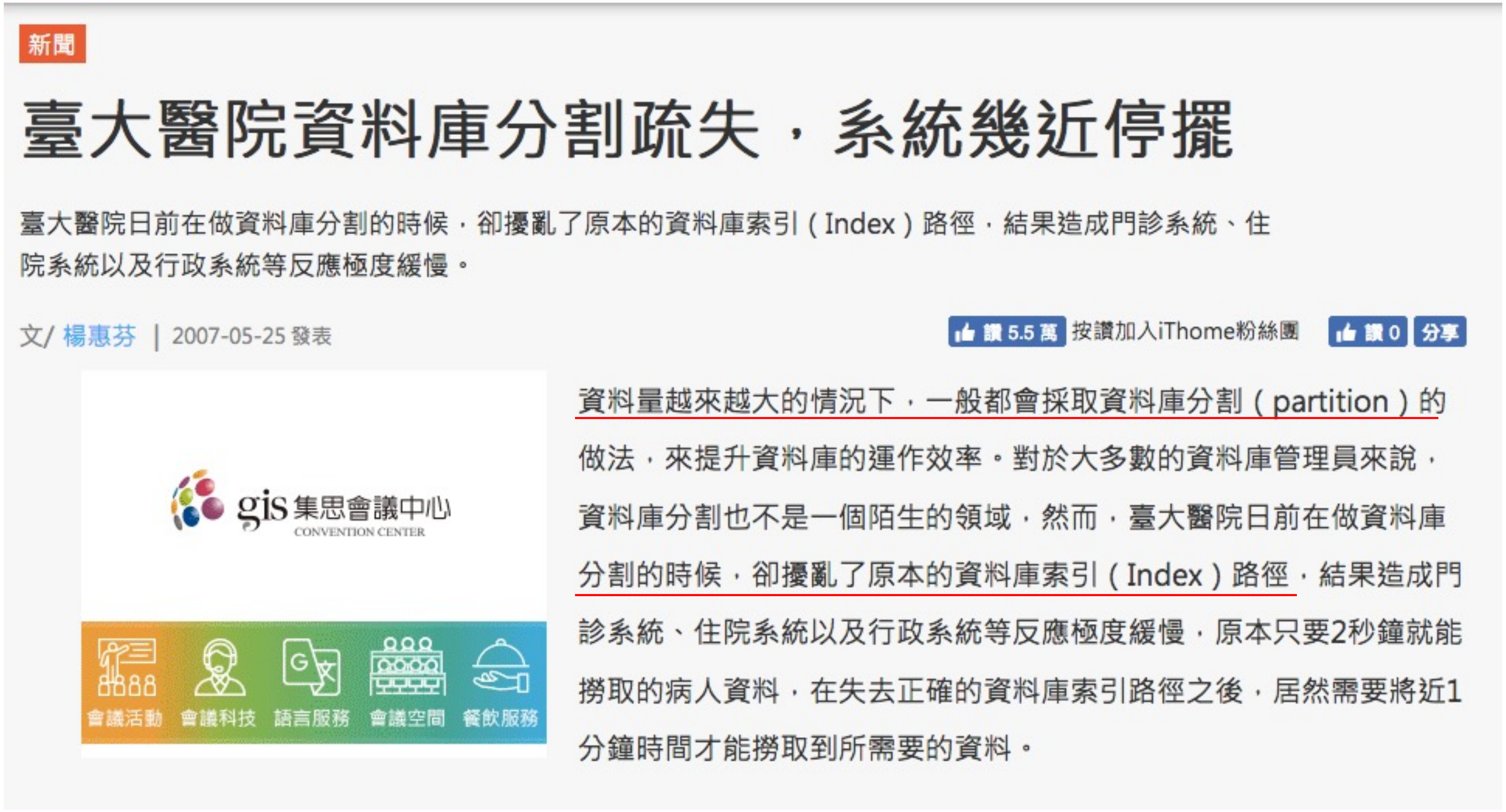
我也沒辦法得知 主機是否有做啥設定之類的

只能來這問問大家 問題到底出在哪

拜託了 各位Q_Q



```
我也沒辦法得知 主機是否有做啥設定之類的

只能來這問問大家 問題到底出在哪

拜託了 各位Q_Q


--
※ 發信站: 批踢踢實業坊(ptt.cc), 來自: 123.0.52.121
※ 文章網址: https://www.ptt.cc/bbs/PHP/M.1487261567.A.08E.html
※ 編輯: liisi (123.0.52.121), 02/17/2017 00:16:21
→ AndCycle: 你這是 Database 版的問題, 加索引會增加寫入負擔              02/17 02:34
→ et69523820: 你可以把索引加回去  看一下記憶體是不是被吃光了           02/17 02:53
→ et69523820: 如果加索引記憶體導致不夠用請檢查key_buffer_size          02/17 02:56
→ et69523820: 你資料庫是用mysql嗎 版本多少 版本如果是5.1那邊的        02/17 02:58
→ et69523820: 會容易有死鎖問題                                         02/17 02:58
→ sonicjr: 索引加太多欄位會反效果 而且才10幾萬筆就慢感覺不太對         02/17 11:44
→ chenxiaowoo: 嗯~幾個欄位? 有text嗎? 如果有建議不要，full text      02/19 12:11
→ chenxiaowoo: 的索引幾十萬筆...不是一般硬體撐得起的                   02/19 12:12
推 IhaveASecret: 做一下 EXPLAIN [Query]                               02/20 20:41
```



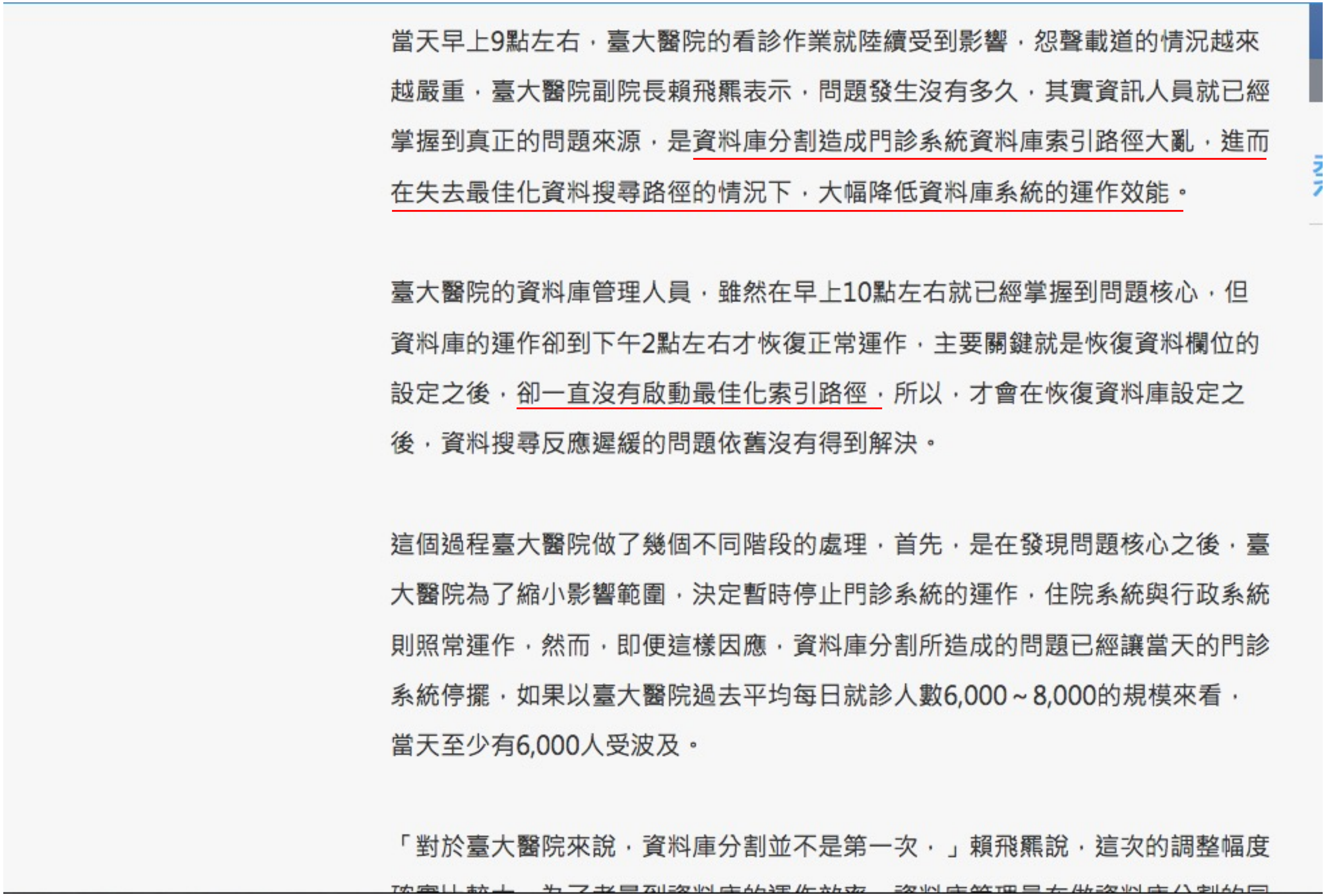
新聞

臺大醫院資料庫分割疏失，系統幾近停擺

臺大醫院日前在做資料庫分割的時候，卻擾亂了原本的資料庫索引（Index）路徑，結果造成門診系統、住院系統以及行政系統等反應極度緩慢。

文/ 楊惠芬 ｜ 2007-05-25 發表

資料量越來越大的情況下，一般都會採取資料庫分割（partition）的做法，來提升資料庫的運作效率。對於大多數的資料庫管理員來說，資料庫分割也不是一個陌生的領域，然而，臺大醫院日前在做資料庫分割的時候，卻擾亂了原本的資料庫索引（Index）路徑，結果造成門診系統、住院系統以及行政系統等反應極度緩慢，原本只要2秒鐘就能撈取的病人資料，在失去正確的資料庫索引路徑之後，居然需要將近1分鐘時間才能撈取到所需要的資料。


當天早上9點左右，臺大醫院的看診作業就陸續受到影響，怨聲載道的情況越來越嚴重，臺大醫院副院長賴飛羆表示，問題發生沒有多久，其實資訊人員就已經掌握到真正的問題來源，是資料庫分割造成門診系統資料庫索引路徑大亂，進而在失去最佳化資料搜尋路徑的情況下，大幅降低資料庫系統的運作效能。

臺大醫院的資料庫管理人員，雖然在早上10點左右就已經掌握到問題核心，但資料庫的運作卻到下午2點左右才恢復正常運作，主要關鍵就是恢復資料欄位的設定之後，卻一直沒有啟動最佳化索引路徑，所以，才會在恢復資料庫設定之後，資料搜尋反應遲緩的問題依舊沒有得到解決。

這個過程臺大醫院做了幾個不同階段的處理，首先，是在發現問題核心之後，臺大醫院為了縮小影響範圍，決定暫時停止門診系統的運作，住院系統與行政系統則照常運作，然而，即便這樣因應，資料庫分割所造成的問題已經讓當天的門診系統停擺，如果以臺大醫院過去平均每日就診人數6,000～8,000的規模來看，當天至少有6,000人受波及。

「對於臺大醫院來說，資料庫分割並不是第一次，」賴飛羆說，這次的調整幅度

Outline

- ♦ Physical Database Design
- ♦ Performance Tuning
  - – Tuning Database Design
  - – Tuning Query
  - – Tuning Index

Physical DB Design

- Physical DB Design: process of
  - choosing specific storage structures & access paths (indexing) for the DB files
  - To achieve good performance for various DB applications
- DBMS offers
  - a variety of options for file organization & access paths
    - clustering of related records on disk blocks
    - linking related records via pointers
    - types of indexing
    - types of hashing

Physical DB Design (cont.)

- Criteria to guide choice of physical DB design
  - Response time: influenced by
    - Under control of DBMS: DB access time for data items referenced by the transaction
    - Not under control of DBMS: system load, OS scheduling, communication delays
  - Space utilization
    - Amount of storage space used by DB files & access path structures on disk
  - Transaction throughput
    - Must be measured under peak conditions on the system

Factors that influence Physical DB Design

- For a given conceptual schema, there are many physical design alternatives in a given DBMS
- Factors that influence physical Database design
  - Database query & transactions
  - Expected frequency of invocation of queries & transactions
  - Time constraints of queries & transactions
  - Expected frequencies of update operations
  - Uniqueness constraints on attributes

Analyzing Database Queries

- Expected queries run on the DB
  - Files that will be accessed by the query
  - Attributes on which selection conditions for the queries are specified
  - Attributes on which join conditions conditions to link multiple tables for the query are specified
  - Attributes whose values will be retrieved by the query
  - Attributes of selection & join conditions are candidates for the definition of index structures

```
SELECT  FNAME, LNAME, ADDRESS
FROM      (EMPLOYEE JOIN DEPARTMENT ON DNUMBER = DNO)
WHERE    DNAME = 'Research’
```

Analyzing Database Transactions

- Expected update transactions run on the DB
  1. Files that will be updated
  2. Type of operation on each file (insert, update, delete)
  3. Attributes on which selection conditions for a delete or update are specified
  4. Attributes whose values will be changed by an update operation

  * Attributes of item 3 are candidates for the definition of index structures

  * Attributes of item 4 are candidates for avoiding an indexing structure

```
UPDATE   PROJECT
SET  PLOCATION = 'Bellaire', DNUM = 5
WHERE    PNUMBER = 10
```

Analyzing Expected Frequency of Invocation of Queries & Transactions

- 80-20 rules
  - Approximately 80% of the processing is accounted for by only 20% of queries & transactions
  - It is sufficient to determine the 20% most important ones

Analyzing Time Constraints of Queries & Transactions

- Selection attributes used by queries & transactions with time constraints become higher-priority candidates for primary index structures

```
SELECT  FNAME, LNAME, ADDRESS
FROM     (EMPLOYEE JOIN DEPARTMENT ON DNUMBER = DNO)
WHERE    DNAME = 'Research’
```

Analyzing Expected Frequencies of Update Operations

- A minimum number of access paths (index) should be specified for a file that is updated frequently

```
UPDATE   PROJECT
SET  PLOCATION = 'Bellaire', DNUM = 5
WHERE    PNUMBER = 10
```

Analyzing Uniqueness Constraints on Attributes

- Access path should be specified on all candidate key attributes that are either the primary key or constrained to be unique
- The existence of an index makes it sufficient to search only the index when checking this constraint, since all values of the attribute will exist in the leaf nodes of the index

Outline

- Physical Database Design
- Performance Tuning
  - Tuning Database Design
  - Tuning Query
  - Tuning Index

Tuning

- Tuning: monitor & revise physical DB design constantly
- Goals of tuning
  - Makes applications run faster
  - Lower response time of queries/transactions
  - Improve overall throughput of transactions

Information Statistics for Tuning

- Number of records in a table
- Number of distinct values in an attribute
- Storage statistics (table space, index space, buffer)
- Number of times a particular query or transaction is submitted /executed in an interval of time
- I/O & device performance statistics
- Query/transaction processing statistics
- Locking/logging related statistics
- Index statistics (number of levels, number of leaves)

Major Tasks of Tuning

- Tuning Database Design
- Tuning Query
- Tuning Indexes

Tuning Database Design

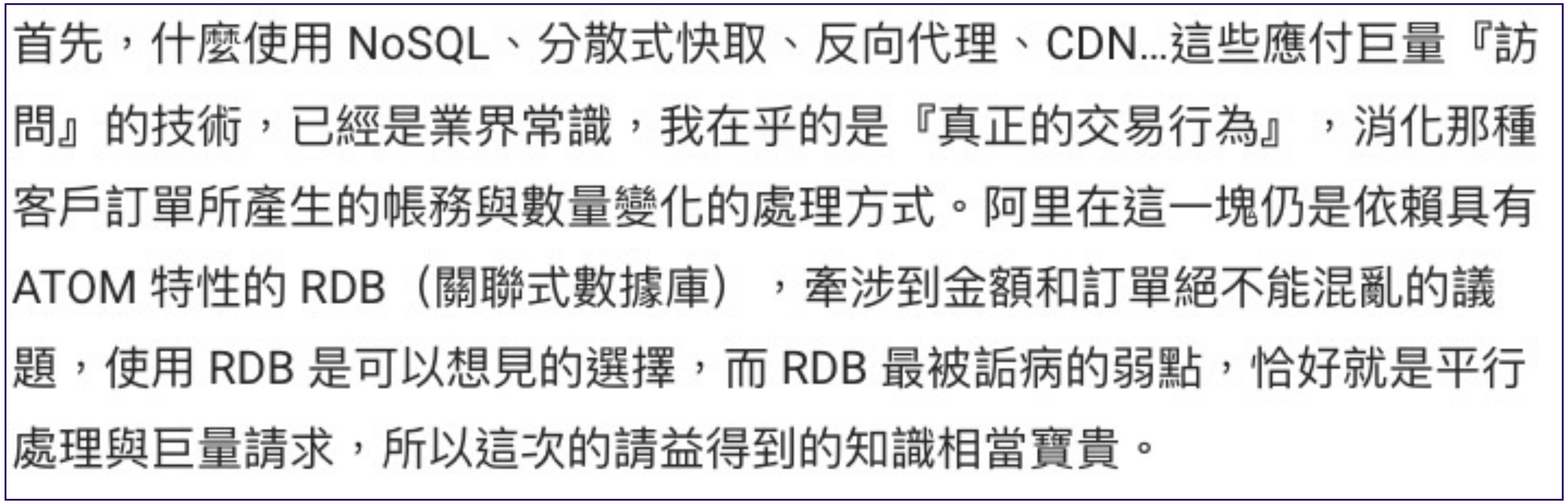
- Existing tables may be joined (denormalized) because certain attributes from tables are frequently needed together
- For the given set of tables, there may be alternative design choices, all of which achieve 3NF or BCNF. One may be replaced by the other
- Vertical partitioning
  - Employee(SSN, Name, Phone, Grade, Salary)
  - Emp1(SSN, Name, Phone), Emp2(SSN, Grade, Salary)
- Horizontal partitioning
  - e.g. product table may be split to 10 tables based on 10 product lines
- Attributes from one table may be repeated in another even though this creates redundancy & potential anomaly

【雙 11 訂單＝超級DDos】阿里巴巴技術團隊如何確保網站不當機？

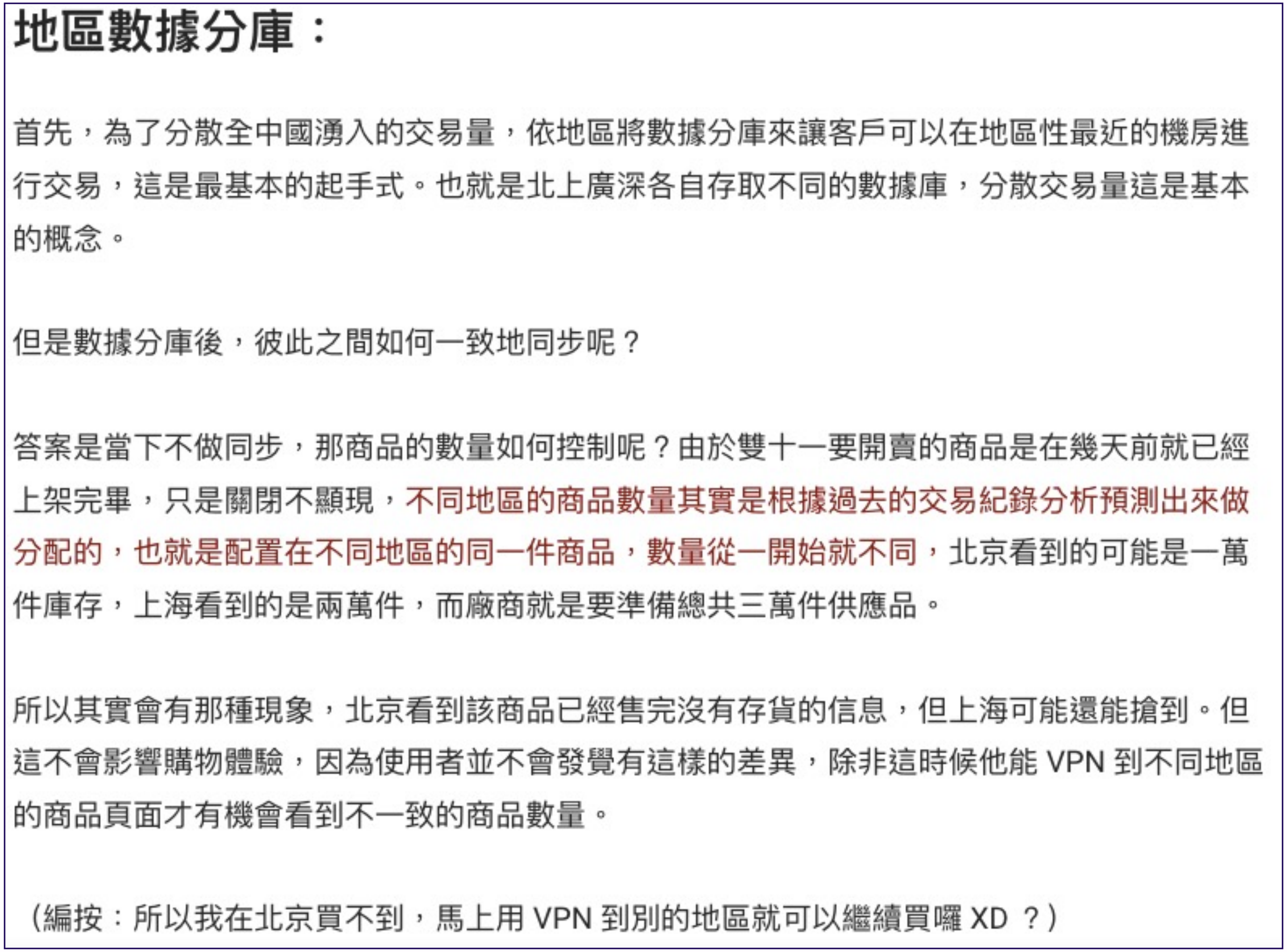
TO TO 精選觀點 2017-10-30

SHARE

Beijing
Hangzhou
Tokyo
hen
Hongkong
24:00:00
2016年11月11日
24:00:00 2016 11.11 Global Shopping Festival total GMV is RMB 120.7 billion.
1207 亿



首先，什麼使用 NoSQL、分散式快取、反向代理、CDN…這些應付巨量『訪問』的技術，已經是業界常識，我在乎的是『真正的交易行為』，消化那種客戶訂單所產生的帳務與數量變化的處理方式。阿里在這一塊仍是依賴具有 ATOM 特性的 RDB（關聯式數據庫），牽涉到金額和訂單絕不能混亂的議題，使用 RDB 是可以想見的選擇，而 RDB 最被詬病的弱點，恰好就是平行處理與巨量請求，所以這次的請益得到的知識相當寶貴。



地區數據分庫：

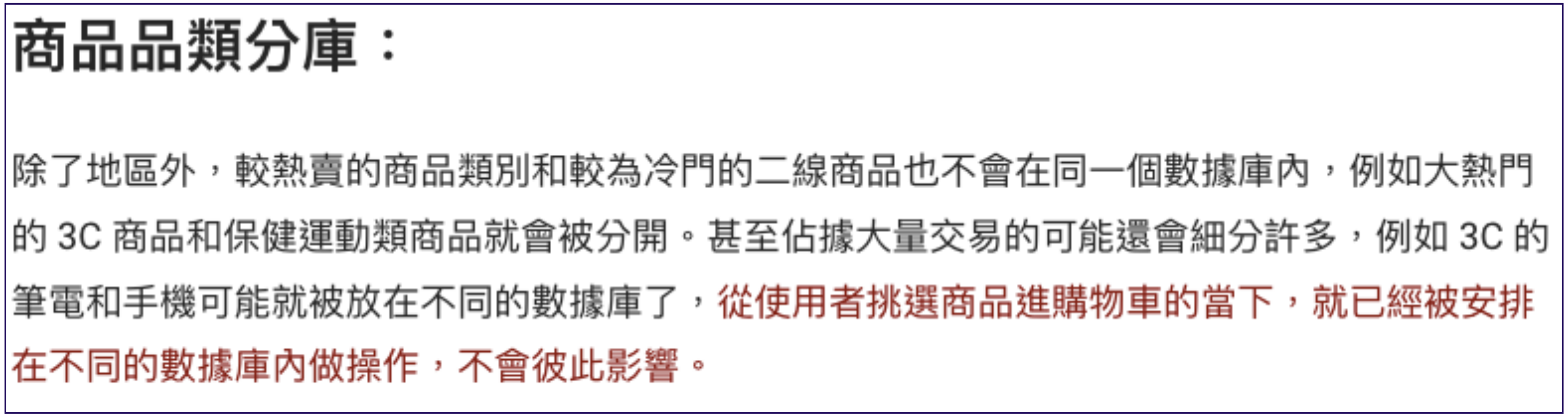
首先，為了分散全中國湧入的交易量，依地區將數據分庫來讓客戶可以在地區性最近的機房進行交易，這是最基本的起手式。也就是北上廣深各自存取不同的數據庫，分散交易量這是基本的概念。

但是數據分庫後，彼此之間如何一致地同步呢？

答案是當下不做同步，那商品的數量如何控制呢？由於雙十一要開賣的商品是在幾天前就已經上架完畢，只是關閉不顯現，不同地區的商品數量其實是根據過去的交易紀錄分析預測出來做分配的，也就是配置在不同地區的同一件商品，數量從一開始就不同，北京看到的可能是一萬件庫存，上海看到的是兩萬件，而廠商就是要準備總共三萬件供應品。

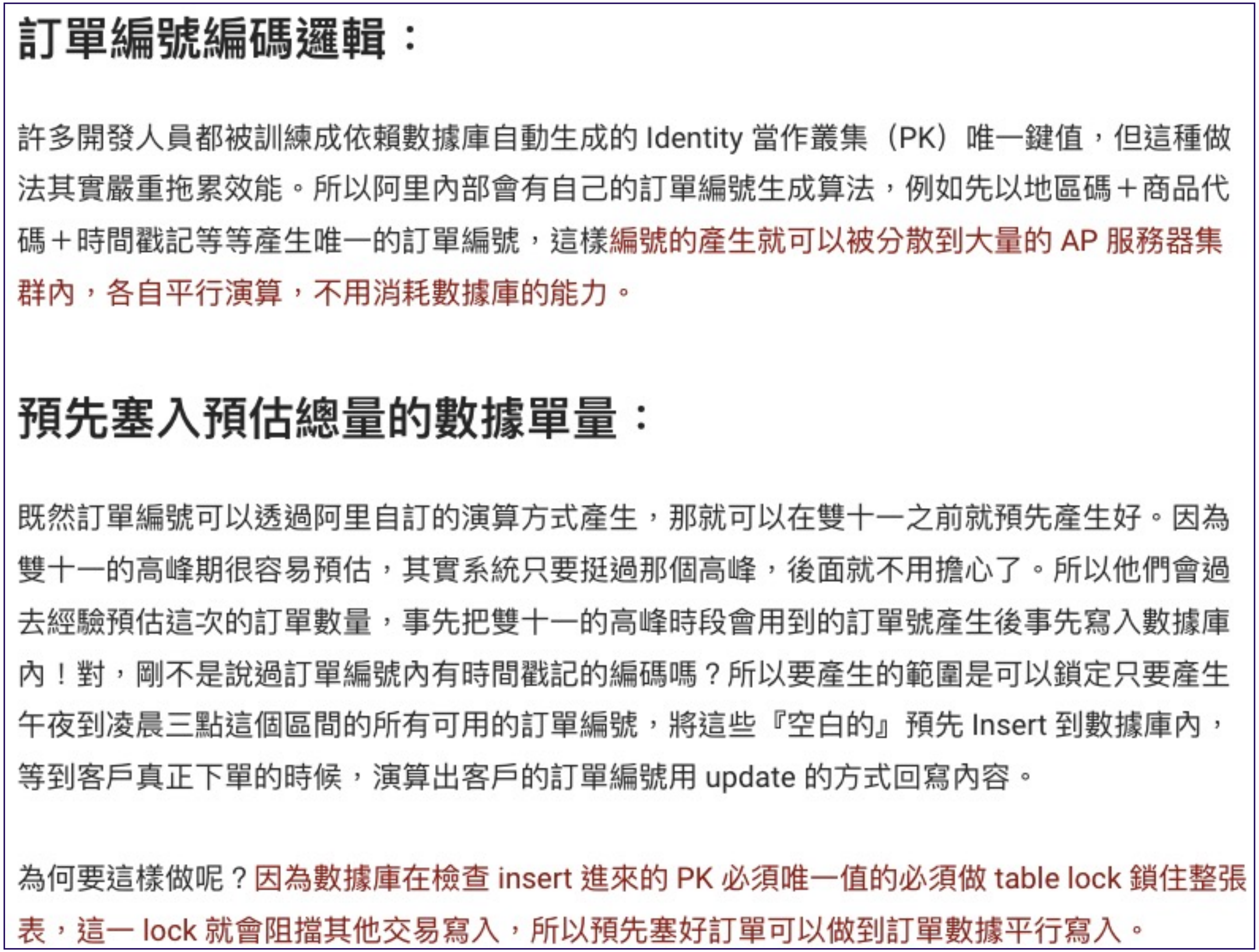
所以其實會有那種現象，北京看到該商品已經售完沒有存貨的信息，但上海可能還能搶到。但這不會影響購物體驗，因為使用者並不會發覺有這樣的差異，除非這時候他能 VPN 到不同地區的商品頁面才有機會看到不一致的商品數量。

（編按：所以我在北京買不到，馬上用 VPN 到別的地區就可以繼續買囉 XD ？）



商品品類分庫：

除了地區外，較熱賣的商品類別和較為冷門的二線商品也不會在同一個數據庫內，例如大熱門的 3C 商品和保健運動類商品就會被分開。甚至佔據大量交易的可能還會細分許多，例如 3C 的筆電和手機可能就被放在不同的數據庫了，從使用者挑選商品進購物車的當下，就已經被安排在不同的數據庫內做操作，不會彼此影響。



訂單編號編碼邏輯：

許多開發人員都被訓練成依賴數據庫自動生成的 Identity 當作叢集（PK）唯一鍵值，但這種做法其實嚴重拖累效能。所以阿里內部會有自己的訂單編號生成算法，例如先以地區碼＋商品代碼＋時間戳記等等產生唯一的訂單編號，這樣編號的產生就可以被分散到大量的 AP 服務器集群內，各自平行演算，不用消耗數據庫的能力。

預先塞入預估總量的數據單量：

既然訂單編號可以透過阿里自訂的演算方式產生，那就可以在雙十一之前就預先產生好。因為雙十一的高峰期很容易預估，其實系統只要挺過那個高峰，後面就不用擔心了。所以他們會過去經驗預估這次的訂單數量，事先把雙十一的高峰時段會用到的訂單號產生後事先寫入數據庫內！對，剛不是說過訂單編號內有時間戳記的編碼嗎？所以要產生的範圍是可以鎖定只要產生午夜到凌晨三點這個區間的所有可用的訂單編號，將這些『空白的』預先 Insert 到數據庫內，等到客戶真正下單的時候，演算出客戶的訂單編號用 update 的方式回寫內容。

為何要這樣做呢？因為數據庫在檢查 insert 進來的 PK 必須唯一值的必須做 table lock 鎖住整張表，這一 lock 就會阻擋其他交易寫入，所以預先塞好訂單可以做到訂單數據平行寫入。

Outline

- Performance Tuning
- Tuning Database Design
- Tuning Query
- Tuning Index

Tuning Query

- Query performance is dependent on appropriate selection of indexes
- Indexes may have to be tuned after analyzing queries that give poor performance
- 2 indications to tune query
  - A query issues too many disk accesses
  - Query plan shows that relevant indexes are not being used

Tips of Tuning Query

- Many query optimizers don't use indexes in the presence of
  - Arithmetic expressions (salary/365 > 10)
  - Numerical comparisons of attributes of different sizes & precision

    AQTY = BQTY, AQTY: integer, BQTY: short integer
  - Null comparisons (Bdate is null)
  - Substring comparisons (Lname like "%mann") (it depends)

Tips of Tuning Query (cont.)

- Indexes are often not used for nested query using IN

```
SELECT SSN
FROM Employee
WHERE Dno IN
      ( SELECT DNumber
        FROM Department
        WHERE MgrSSN = '333445555')
```

```
SELECT SSN
FROM    Employee, Department
WHERE DNo=Dnumber AND MgrSSN='333445555'
```

Tips of Tuning Query (cont.)

- A query with multiple selection conditions that are connected via OR may not be prompting query optimizer to use any index

```
SELECT Fname, Lname, Salary, Age
FROM Employee
WHERE Age > 45 OR Salary < 50000;
```

```
SELECT Fname, Lname, Salary, Age
FROM    Employee
WHERE Age > 45
UNION
SELECT Fname, Lname, Salary, Age
FROM Employee
WHERE Salary < 50000;
```

Tuning Query (cont.)

- WHERE conditions may be rewritten to utilize the multi-attribute index (composite index)

```
SELECT Region, Product, Month, Sales
FROM Sales
WHERE Region = 3 AND
            ( (Product BETWEEN 1 AND 3) OR
              (Product BETWEEN 8 AND 10)  );
```

```
SELECT Region, Product, Month, Sales
FROM Sales
WHERE ( (Region = 3) AND (Product BETWEEN 1 AND 3))
              OR
              ( (Region = 3 ) AND (Product BETWEEN 8 AND 10));
```

Tips of Tuning Query (cont.)

- Some DISTINCTs may be redundant & can be avoided without changing the result

  * DISTINCT often causes a sort operation

Tips of Tuning Query (cont.)

- Unnecessary use of temporary result tables
  can be avoided by collapsing multiple queries
  into a single query
  unless temporary relation is needed
  for some intermediate processing

Tips of Tuning Query (cont.)

- In some situations involving use of correlated queries, temporary result tables are useful

```
SELECT SSN
FROM     Employee E
WHERE  Salary = SELECT Max (Salary)
                          FROM Employee M
                          WHERE M.DNo = E.Dno;
```

```
SELECT Max (Salary) AS HighSalary, DNo INTO Temp
 FROM Employee
 GROUP BY Dno;

 SELECT SSN
 FROM Employee, Temp
 WHERE Salary = HighSalary AND Employee.DNo=Temp.Dno;
```

Tips of Tuning Query (cont.)

- If multiple options for join condition are possible, choose one that uses a clustering index &
  avoid those that contain string comparison

e.g. assuming both Name & SSN are candidate key
in Employee & Student,
it is better to use
Employee.SSN = Student.SSN
rather than
Employee.Name = Student.Name
if SSN has a clustering index in one or both tables

Tips of Tuning Query (cont.)

- Try the following transformations
  - NOT condition may be transformed into a positive expression
  - Embedded SELECT blocks using IN, = ALL, = ANY, may be replaced by JOIN
  - If an equality join is set up between 2 tables,

    range predicate (selection condition)

    on the joining attribute set up in one table

    may be repeated for the other table

Tips of Tuning Query (cont.)

- In some query optimizers, order of tables in the FROM clause may affect the join processing
  - switch the order so that
    the smaller of the two relations is scanned and
    the largest relation is used with an appropriate index

R

3	t
5	u
10	v
12	w
18	x
23	y

S

3	A
8	B
10	C
10	D
12	E
17	F
23	G
26	H

Tips of Tuning Query (cont.)

- Some query optimizers perform worse on nested queries compared to equivalent un-nested counterparts
  - (1) Uncorrelated subqueries with aggregates in inner query
  - (2) Uncorrelated subqueries without aggregates
  - (3) Correlated subqueries with aggregates in inner query
  - (4) Correlated subqueries without aggregates

```
SELECT SSN
FROM    Employee E
WHERE  Salary = SELECT Max (Salary)
                FROM Employee M
                WHERE M.DNo = E.Dno;
```

Tips of Tuning Query (cont.)

- Sometimes, views becomes an overkill
- A query may be posed directly against a base table, rather than going through a view defined by a join

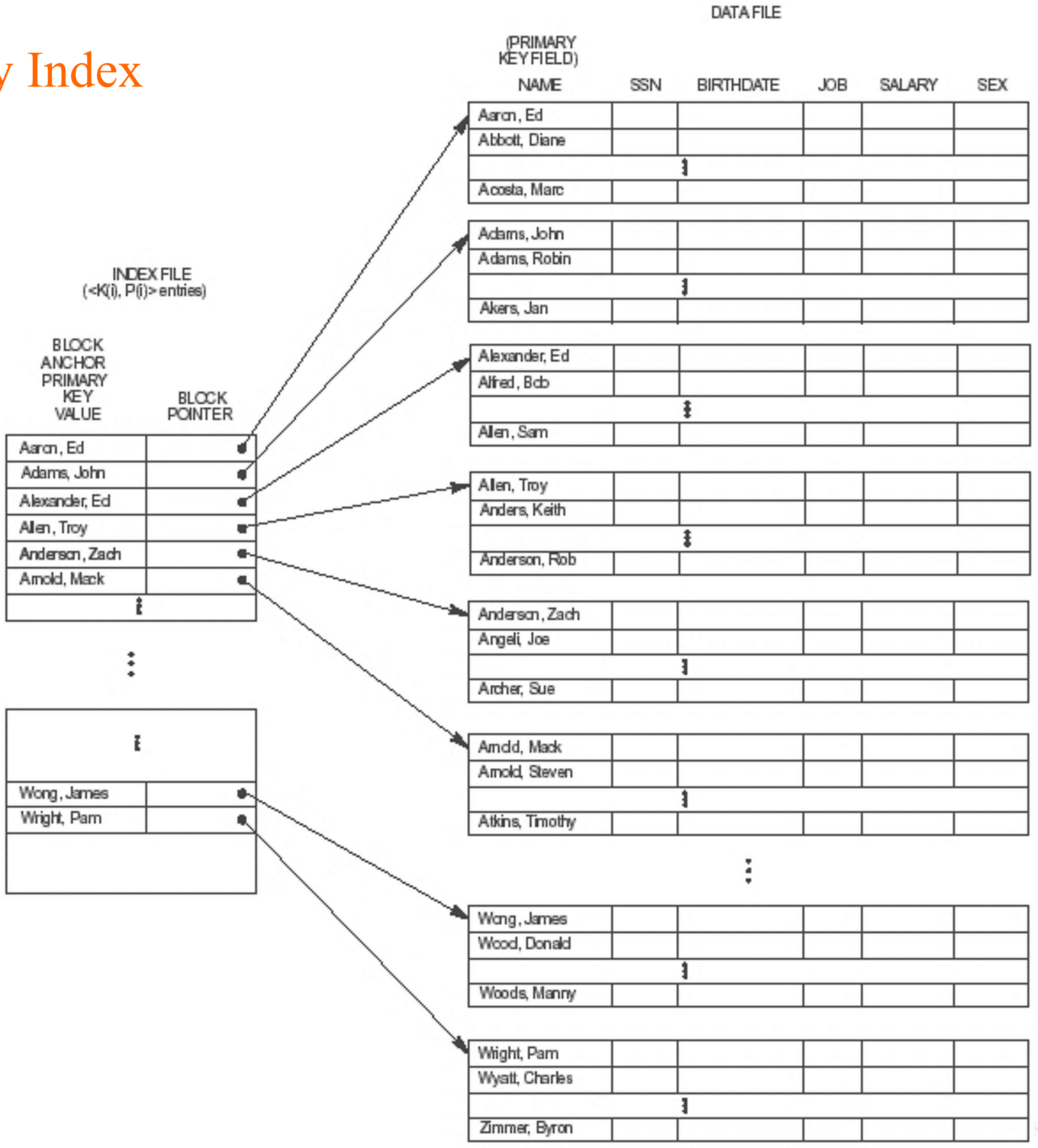
```
CREATE VIEW     WORKS_ON1  AS
SELECT   FNAME, LNAME, PNAME, HOURS
FROM     EMPLOYEE, PROJECT, WORKS_ON
WHERE    SSN=ESSN AND PNO=PNUMBER ;

SELECT  PNAME, FNAME, LNAME
FROM    WORKS_ON1
WHERE   PNAME='ProjectX' ;
```

```
SELECT   FNAME, LNAME, PNAME
FROM     EMPLOYEE, PROJECT, WORKS_ON
WHERE    SSN=ESSN AND PNO=PNUMBER
          PNAME='ProjectX' ;
```

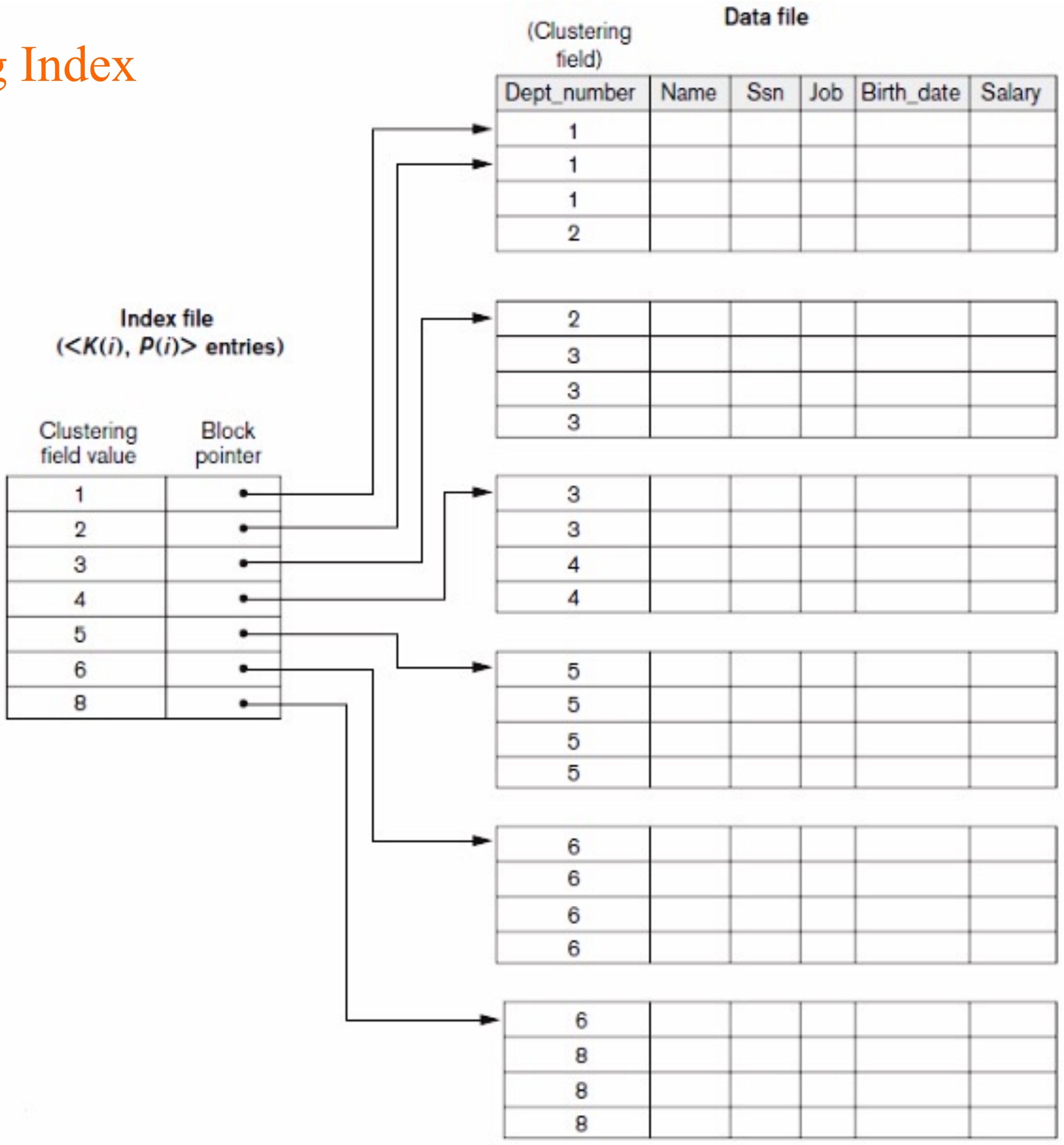
Outline

- Physical Database Design
- Performance Tuning
  - Tuning Database Design
  - Tuning Query
  - Tuning Index

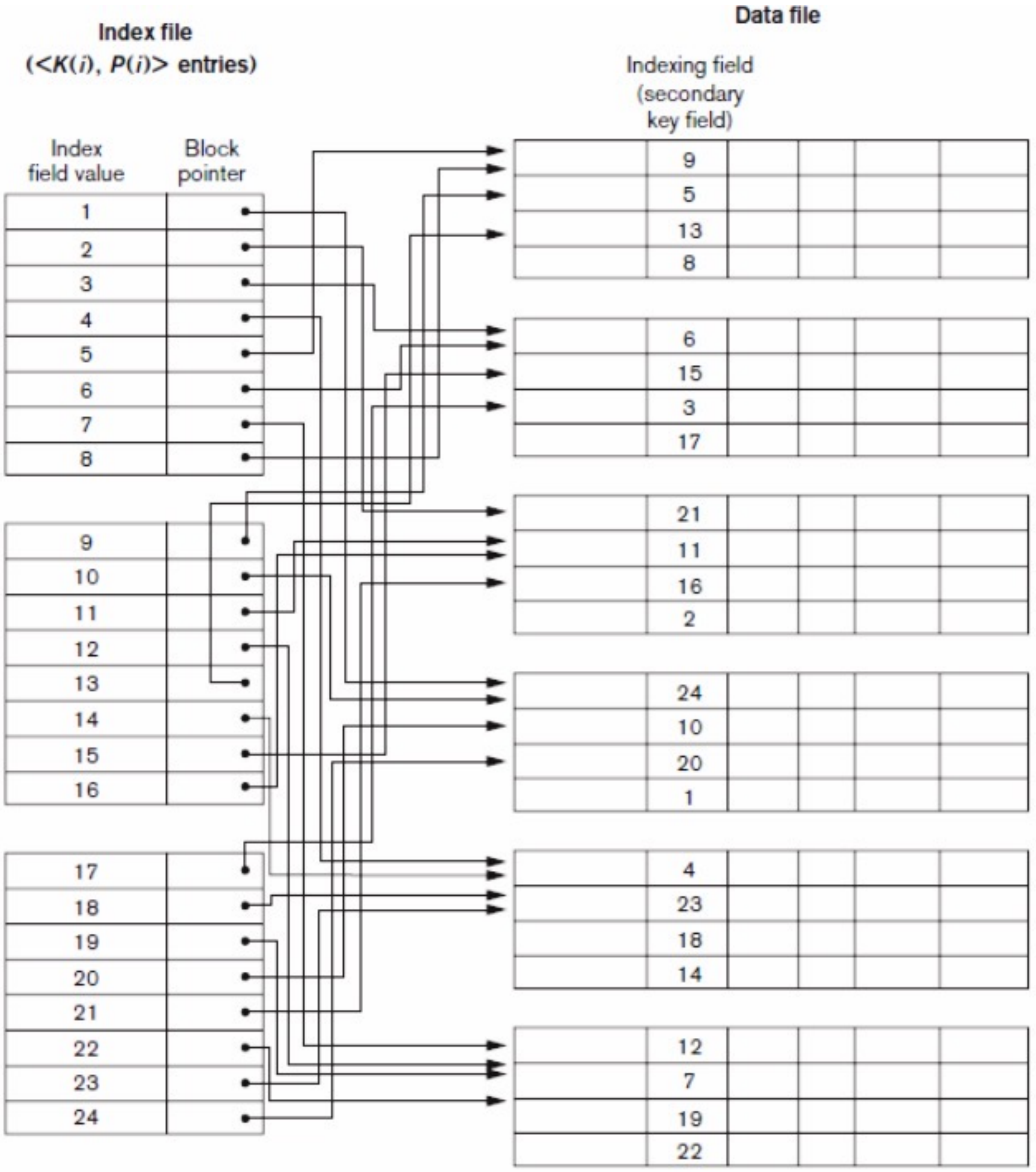
Primary Index



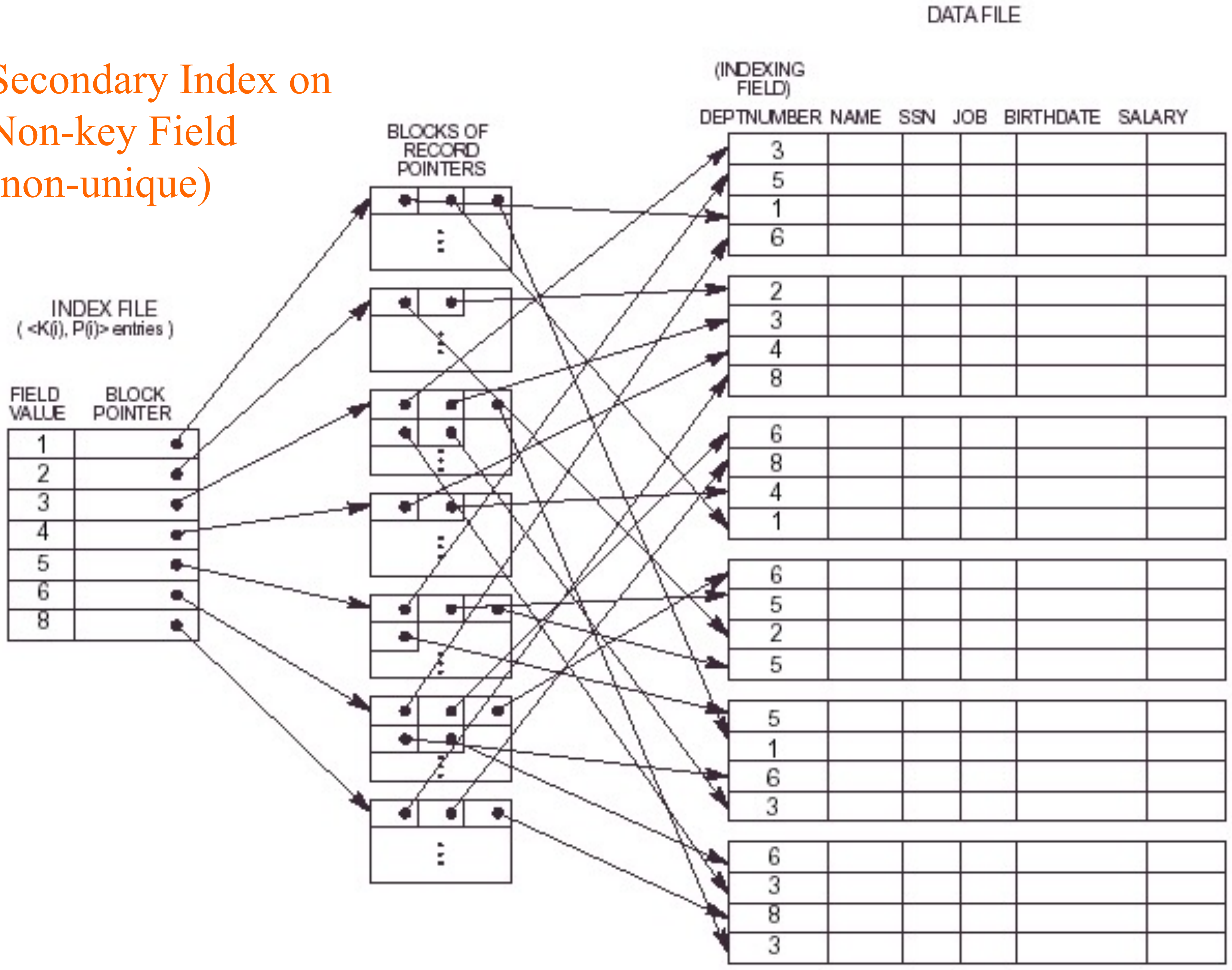
Clustering Index



Secondary Index on Key Field (unique)



Secondary Index on Non-key Field (non-unique)



Design Decisions about Indexing

- Whether to index an attribute
  - Attribute is a key or used by a query (selection condition) or join attribute
- What attribute(s) to index on
  - Single or multiple
- Whether to set up a clustered index
  - One per table
- Whether to use a hash index over a tree index
  - Some hash indexes do not support range queries
- Whether to use dynamic hashing
  - For files that grow & shrink continuously

Tuning Indexes

- Reasons for tuning indexes
  - Certain queries may take too long to run for lack of an index
  - Certain indexes may not get utilized at all
  - Certain indexes may be causing excessive overhead because index is on an attribute that undergoes frequent changes

Tuning Indexes (cont.)

- Goals of Tuning Index
  - To dynamically evaluate the requirements
  - To drop a index or to build a new index
  - To reorganize indexes to yield the best overall performance

Consideration of Tuning Index

- Dropping & building indexes is an overhead
- Updating table is suspend while an index is dropped or created
- Rebuilding index may improve performance
  - Too many deletion on B+tree produces waste space
  - Too many deletion on B+Tree causes overflow space
- Rebuilding a clustered index amounts to reorganizing the entire table ordered on that key
- Limitation of options for indexes from system to system
  - Sybase: sparse clustering index as B+tree
  - INGRES: sparse clustering index as ISAM files
  - Oracle, DB2: clustering index is limited to dense index

Conclusions

- Performance Tuning
- Tuning Database Design
- Tuning Query: rewritten query
  - Utilize index
  - Avoid unnecessary disk I/O (temporary table)
  - JOIN
  - VIEW
- Tuning Index

Conclusions

- Performance Tuning
- Tuning Database Design
- Tuning Query: rewritten query
  - Utilize index
  - Avoid unnecessary disk I/O (temporary table)
  - JOIN
  - VIEW
- Tuning Index