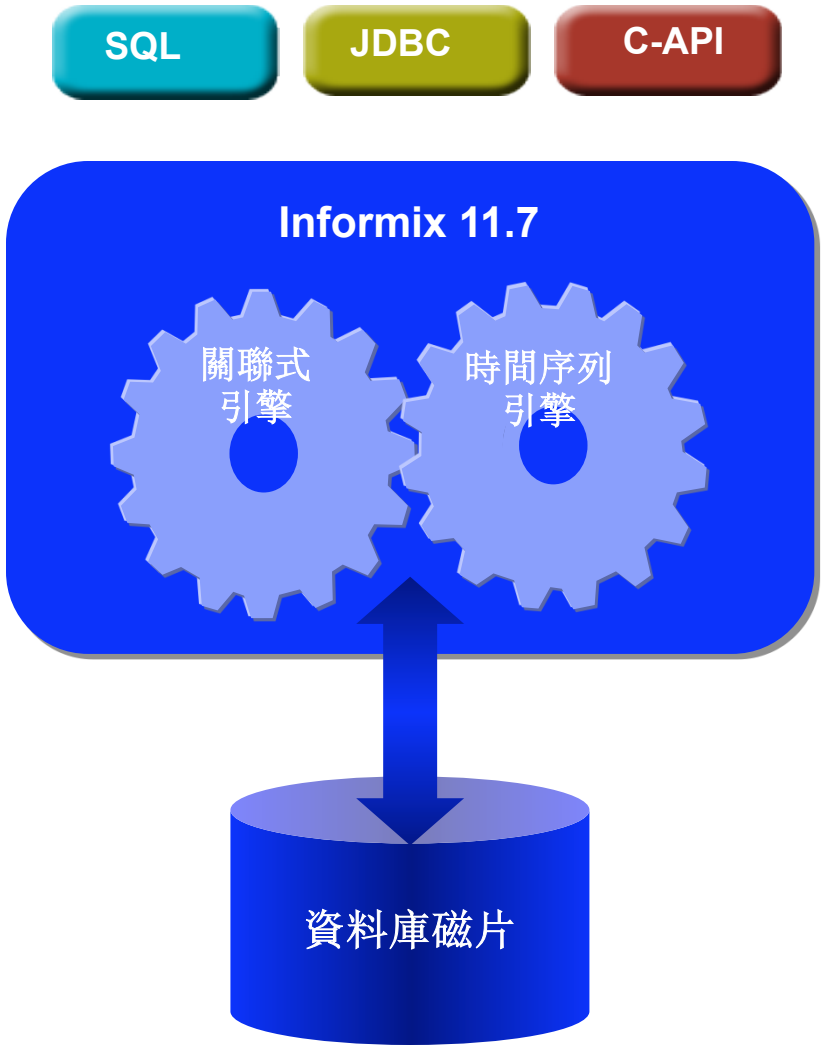
# Informix 11.7時間序列資料庫

Informix. software

# 議程

- Informix 11.7時間序列(TimeSeries)引擎
- 什麼是時間序列?
- 關鍵概念
- 創建時間序列應用
- 為什麼需要即時裝載器?
- 即時裝載器概要
- 匯總

# Informix 11.7含時間序列(Timeseries)引擎




- 日期戳和時間戳記資料正在以驚人的速度不斷增長
- Informix 11.7雙引擎資料庫
- 時間序列引擎突破了當前關係型數據庫“海量電錶資料收集”的瓶頸
- Informix資料庫裡關聯式與時間序列(TimeSeries) 型 互補共存，滿足客戶多樣性的需求。

# 什麼是時間序列?

- 時間序列資料是:
  - 每個資料都有時間戳記得一組資料集
    - 可以想像為一個陣列，它的每個元素可以被時間或者時間戳記索引

      “在時間序列”X“中把1月1日的數據給我”

- 當一段範圍內的資料被順序讀取時效率最高

  “在時間序列”X“中把1月1日到1月10日的數據給我”

- 通常情況下，在完全讀取一個時間序列資料後，才去讀取下一個時間序列。

# 誰在用時間序列資料?

- 華爾街
  - 金融市場數據
- 性能監視
  - 智能電網
  - 網路設備
  - 製造工藝流程
  - 隨時間變化的溫度
- 趨勢分析
  - 智能電網–節假日用電分析
  - 網路設備錯誤分析
- 基於位置的服務
  - 隨時間變化的卡車車隊的位置
  - RFID（射頻識別技術）
- 還有其它很多...

# 這些市場的共同點是什麼?

- 通過時間段訪問資料
  - 關注過去的一段時間範圍的資料
  - 對未來一段時間範圍進行預測
- 這種分析很具有針對性
- 需要保存大量的線上資料
- 每秒鐘都會有巨量的資料參與
- 所有這些市場都使用關聯式資料
- 都需要把關聯式資料與時間序列資料合併

# 提示:

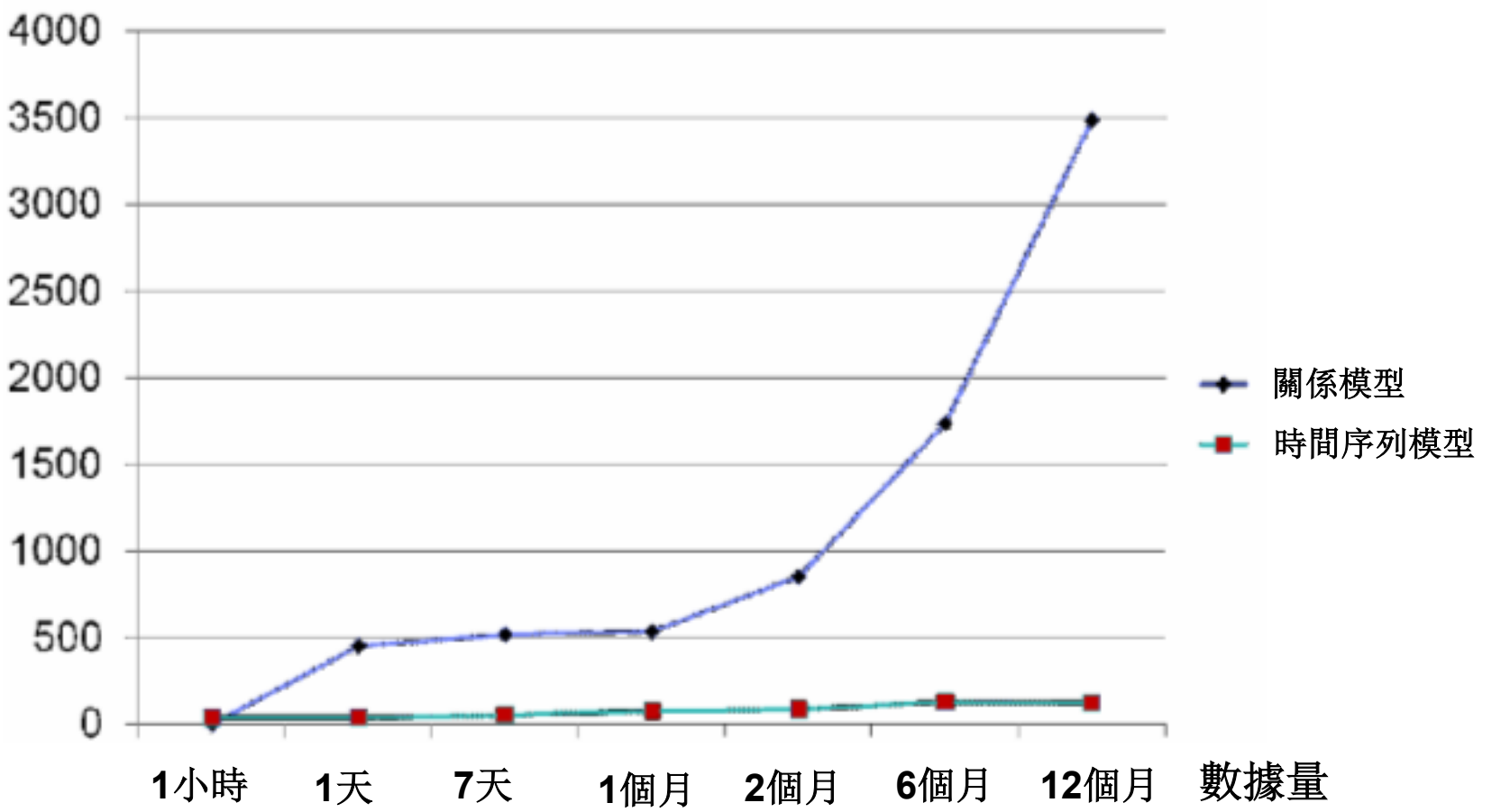
- 一定範圍的資料查詢結合聚合資料效率最佳
- 巨量的資料要求更高的性能
  - 智能電網– 10,000,000電錶計算每日用電量3分鐘?
- 需要使用**SQL**語句存取時間序列數據
- 需要一定的方法來寫自己的分析

# Informix時間序列的主要優點

- 性能
  - 非常快速的資料讀取
    - 資料在磁片上被優化存儲
  - 使用標準SQL操作非常困難或者不可能
- 節省存儲空間
  - 在最好的情況下，可以節省50%以上的存儲空間
- 使用工具包允許使用者開發他們自己的演算法
  - 演算法利用資料庫緩衝集區運行
- 在概念上更貼近用戶是如何想像時間序列的

# Informix時間序列與關聯式的比較

執行時間（秒）



# 關鍵概念:固定(Regular)的時間序列

- 在固定時間間隔收集的資料形成一個固定時間的時間序列
  - 例如:每天,每小時,等等...
- 在某些時間點上可以沒有資料
  - 對於一個美國的工作周來說，從週一到週五收集資料，星期六和星期日不收集
  - 智能電網7 * 24 * 365監控
  - 網路設備7 * 24 * 365監控
- 一個時間序列在一個時間點最多只能有一個資料
  - 對於特定的時間點沒有資料是可接受的
- 一個時間的資料不會延續到下一個
- 可以被看作是一個陣列

# 關係模型架構1 -智慧電網 或 網路設備

| CP_NO | Data_date | Data_type | KWH_ reading | KWH |
|---|---|---|---|---|
| 0001 | 2010-01-01 | … | … | 1.00 |
| 0001 | 2010-01-02 | … | … | 1.01 |
| 0001 | 2010-01-03 | … | … | 1.02 |
| 0001 | 2010-01-04 | … | … | 1.03 |
| …. | | … | … | |
| 0002 | 2010-01-01 | … | … | 1.10 |
| 0002 | 2010-01-02 | … | … | 1.11 |
| 0002 | 2010-01-03 | … | … | 1.12 |
| …. | 2010-01-04 | … | … | 1.14 |
| 0003 | 2010-02-01 | … | … | 1.25 |
| 0003 | 2010-02-02 | … | … | 1.26 |
| 0003 | … | … | … | … |

關係模型：

24 * 365 * 100,000

= 876,000,000

- 100,000戶每小時採集一次，累計365天
- 關聯式資料表格及達－ 876,000,000行數據

# 時間序列模型(Informix) -智慧電網 或 網路設備

每一行有24 * 365 = 8,760個對象

| CP_NO | Time Series Data 24 * 365 |
|---|---|
| 0001 | (2010-01-01--00:00, 1.00), (2010-01-01-00:15, 1.02),(2010-01-01-01:30, 1.02), …. |
| 0002 | (2010-01-01--00:00, 1.00), (2010-01-01-00:15, 1.02),(2010-01-01-01:30, 1.02), …. |
| 0003 | (2010-01-01--00:00, 1.00), (2010-01-01-00:15, 1.02),(2010-01-01-01:30, 1.02), …. |
| 0004 | (2010-01-01--00:00, 1.00), (2010-01-01-00:15, 1.02),(2010-01-01-01:30, 1.02), …. |
| …. | |
| 1001 | (2010-01-01--00:00, 1.10), (2010-01-01-00:15, 1.12),(2010-01-01-01:30, 1.13), …. |
| 1002 | (2010-01-01--00:00, 2.20), (2010-01-01-00:15, 2.22),(2010-01-01-01:30, 2.22), …. |
| 1003 | (2010-01-01--00:00, 3.30), (2010-01-01-00:15, 3.33),(2010-01-01-01:30, 3.33), …. |
| …. | |
| 2001 | (2010-01-01--00:00, 4.10), (2010-01-01-00:15, 5.12),(2010-01-01-01:30, 5.13), …. |
| 2002 | (2010-01-01--00:00, 4.20), (2010-01-01-00:15, 5.22),(2010-01-01-01:30, 5.22), …. |

時間序列：

100,000行

- 100,000戶每小時採集一次，累計365天
- 時間序列資料表格只有－ 100,000行數據

# 關鍵概念:非固定(Non-Regular)的時間序列

- 對於一個任意的時間間隔可以存儲多於一個資料的時間序列
  - 在一個時間間隔內沒有資料是可以的
- 沒有丟失資料的概念
- 對於非固定的時間序列，一個資料持續多久可以有兩種選擇:
  - 資料可以持續到下一個資料出現
    - 股票價格
  - 資料只在他們出現的時間點有效，並不持續
    - 心跳
- 可以看作是一個資料陣列

# 美國股票價格儲存-關係型數據庫

## 美國股票價格關聯式

| 股票 | 日期時間 | 交易號碼 | 交易價格 | 交易量 |
|---|---|---|---|---|
| I BM | 20100103:0930 | 20100103090301 | 162.55 | 100 |
| IBM | 20100103:0931 | 20100103090302 | 162.65 | 100 |
| IBM | 20100103:0932 | 20100103090303 | 162.75 | 100 |
| IBM | 20100103:0933 | 20100103090304 | 162.85 | 100 |
| IBM | 20100103:0934 | 20100103090305 | 162.95 | 100 |
| | … | | | |
| AAPL | 20100103:0931 | 20100103090321 | 350.65 | 100 |
| AAPL | 20100103:0932 | 20100103090322 | 351.65 | 200 |
| AAPL | 20100103:0933 | 20100103090323 | 350.75 | 100 |
| AAPL | 20100103:0934 | 20100103090324 | 352.65 | 400 |
| AAPL | 20100103:0936 | 20100103090325 | 350.95 | 100 |
| AAPL | 20100103:0937 | 20100103090326 | 350.25 | 500 |
| | …. | | | |

每天數千萬或數億筆交易

# 美國股票價格儲存時間序列型

## 美國股票價格儲存時間序列型

| 股票 | 股票價格儲存時間序列型 |
|---|---|
| IBM | (2010-01-01--00:00, value1), (2010-01-01-00:15, value2),(2010-01-01-01:30, value3), …. |
| ORCL | (2010-01-01--00:00, value1), (2010-01-01-00:15, value2),(2010-01-01-01:30, value3), …. |
| ATT | (2010-01-01--00:00, value1), (2010-01-01-00:15, value2),(2010-01-01-01:30, value3), …. |
| JPM | (2010-01-01--00:00, value1), (2010-01-01-00:15, value2),(2010-01-01-01:30, value3), …. |
| BOA | (2010-01-01--00:00, value1), (2010-01-01-00:15, value2),(2010-01-01-01:30, value3), …. |
| CSCO | (2010-01-01--00:00, value1), (2010-01-01-00:15, value2),(2010-01-01-01:30, value3), …. |
| AAPL | (2010-01-01--00:00, value1), (2010-01-01-00:15, value2),(2010-01-01-01:30, value3), …. |
| GOOG | (2010-01-01--00:00, value1), (2010-01-01-00:15, value2),(2010-01-01-01:30, value3), …. |
| 。 。 。 | 。 。 。 。 |

5000 種股票

# 關鍵概念:日曆

- 每個時間序列對應有一個日曆
- 日曆確定有效的收集資料的時間間隔
- 每個日曆有:
  - 一個開始時間參考
  - 一個 “on”和 “off”模式
    - 模式在開始時間處啟動
    - 數據在 “on”的時間段內被收集
  - 時間間隔定義“on”和 “off”時間段的長度
    - 年,月,日,小時,分鐘,秒

# 實施

# 使用時間序列資料類型

- 時間序列資料類型出現在**Informix 9.20**和其以後的版本中
- 為了使用時間序列資料類型，還需要定義一些額外的資料類型和表:
  - 日曆資料類型(Calendar Datatype)
    - 定義何時開始收集資料
  - 行類型（Row Types）
    - 定義收集什麼樣的資料
  - 容器(container）
    - 定義資料存儲在何處
  - 其它需要的表:
    - Calendar, tsinstancestable, tscontainertable

# 日曆資料類型（Calendar Datatype）

- 使用下面語句創建日曆資料類型:

```
insert into CalendarTable(c_name,c_calendar) values
('c_hour','startdate(2009-01-01 00:00:00.00000),
pattstart(2009-01-01 00:00:00.00000),pattname(cp_hour)');
```

- **Calendartable**是存放時間序列資料類型使用的日曆的表
- **C_NAME**是日曆的名稱
- **C_CALENDAR**包含這個日曆的資訊
  - Start是開始時間參考
  - 模式定義何時收集資料

# 關鍵概念:行類型（Row Types）

- 時間序列的每個片被表示為一行
- 創建行類型的**SQL**語法是:

```
create row typedc_type(
data_date datetime year to fraction(5),
kwh decimal(12,4),
value1 decimal(12,4),
value2 decimal(12,4));
```

  - 注意:時間序列的第一列要求是 “datetime year to fraction(5)”
- 其它列可以是任何類型，但是下面列出的除外:
  - blob, clob, text, serial
- 所有**Informix**對表的正常約束對**columns**數目和**row size**適用
- 時間序列的行（**Rows**）有時候被稱為元素（**elements**）

# 關鍵概念:行類型（Row Types）2

- 定義一個帶有**TimeSeries**的表**dc_table**。由於表的**header**部分只存儲有電錶的基本資訊（記
- 錄數為電錶的數量），故表可以不進行分片。同時為表在欄位**cp_no**上創建一個唯一索引。

```
create table dc_table(
cp_no varchar(16) not null,
org_no varchar(16) not null,
location varchar(255),
data Timeseries(dc_type)
)in tsdbs01 extent size 10240 next size 102400 lock mode row;
create unique index idx_dc_table on dc_table(cp_no);
```

# 關鍵概念:容器（Containers）

- 容器（Containers）是存儲一個或多個時間序列資料的資料機構的名字
- 有“固定”和“非固定”容器
  - 不能把“固定時間序列”和“非固定時間序列”的資料一起存儲在同一個容器內
- 在資料庫空間（dbspace）內創建容器的SQL如下:

  execute procedure TsContainerCreate( ‘cont_name’ , ‘dbspace_name’ , ‘rowtype_name’ , first_extent_size, next_extent_size);

  - Rowtype_name是已經存在的行類型的名字
  - DBSPACE_NAME是已經存在的表空間的名字
  - FIRST_EXTENT_SIZE是第一個extent的大小
  - NEXT_EXTENT_SIZE是接下來的extent的大小

```
execute procedure tscontainercreate ('cn01','tsdbs01','dc_type',1024000,102400);
execute procedure tscontainercreate ('cn02','tsdbs02','dc_type',1024000,102400);
```

- 容器對時間序列資料進行聚合來優化範圍查詢

# 匯總

- 創建日曆（**Create a calendar**）：
  ***insert into Calendartable (c_name, c_calendar) values ('daily', 'start(2001-01-01 00:00:00), pattern({5 on, 2 off}, day)');***
- 創建行類型（**Create a row type**）：

  ***create row type daybar (tstamp datetime year to fraction(5), bid float, ask float, bidsize int, asksize int);***

- 創建容器（**Create a container**）：
  ***execute procedure TsContainerCreate('stock_data', 'stock_parition', 'daybar', 1 GB, 1GB);***
- 創建一個表並插入一行資料（**Create a table and insert a row**）：

  ***create table S_P_500(name char(32), series timeseries(daybar));***
  ***insert into table S_P_500 values ("IBM", "calendar(daily), container(stock_data), regular");***

# 時間序列模型

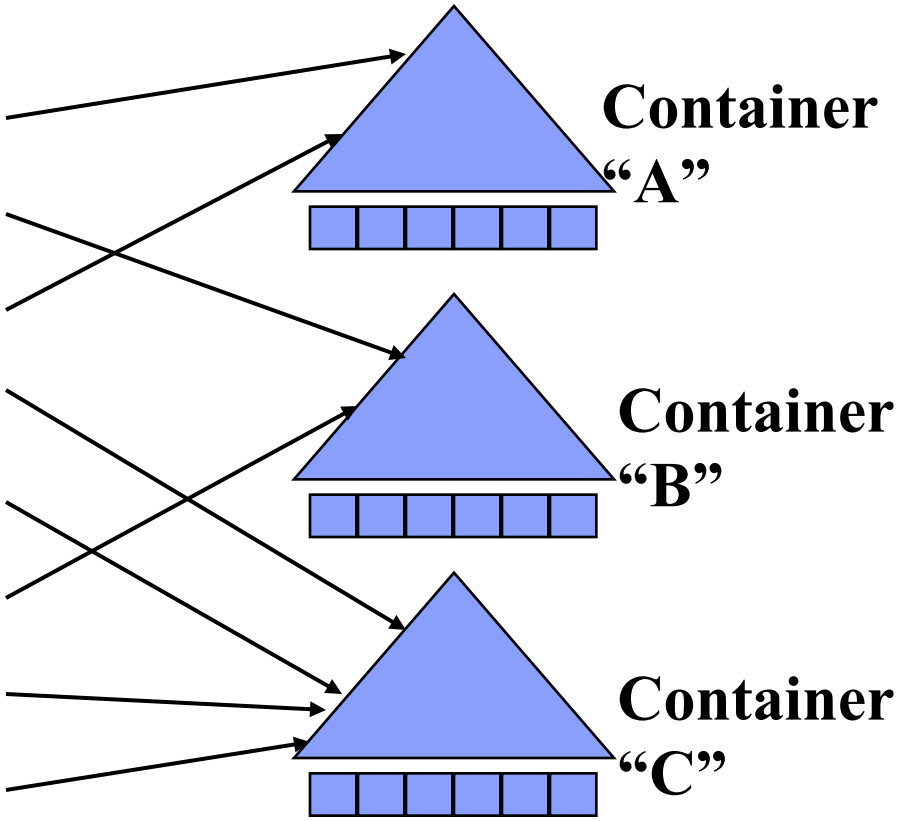
```
insert into CalendarPatterns values('cp_hour','{1 on},hour');
insert into CalendarTable(c_name,c_calendar)
values('c_hour','startdate(2009-01-01 00:00:00.00000),pattstart(2009-01-01 00:00:00.00000),pattname
    (cp_hour)');          /*每小時採集一次資料*/

create row typedc_type(
 data_date datetime year to fraction(5),
 data_type smallint not null,
 fee_flag smallint not null,
 get_date date not null,
 col_time date,                    /*終端抄表時間*/
 kwh_reading decimal(12,4),        /*電能示數*/
 max_need decimal(8,4),            /*最大需量*/
 max_need_time date,               /*最大需量發生時間*/
 kwh decimal(12,4)                 /*電能量*/
);

create table dc_table(
 cp_no varchar(16) not null,
 meapoint_no smallint not null,
 org_no varchar(16) not null,
 dbspace       varchar(128),
 container     varchar(128),
 data Timeseries(dc_type)
)in tsdbs01 extent size 10240 next size 102400 lock mode row;
```

# 一個Informix的表，它包含一列時間序列資料

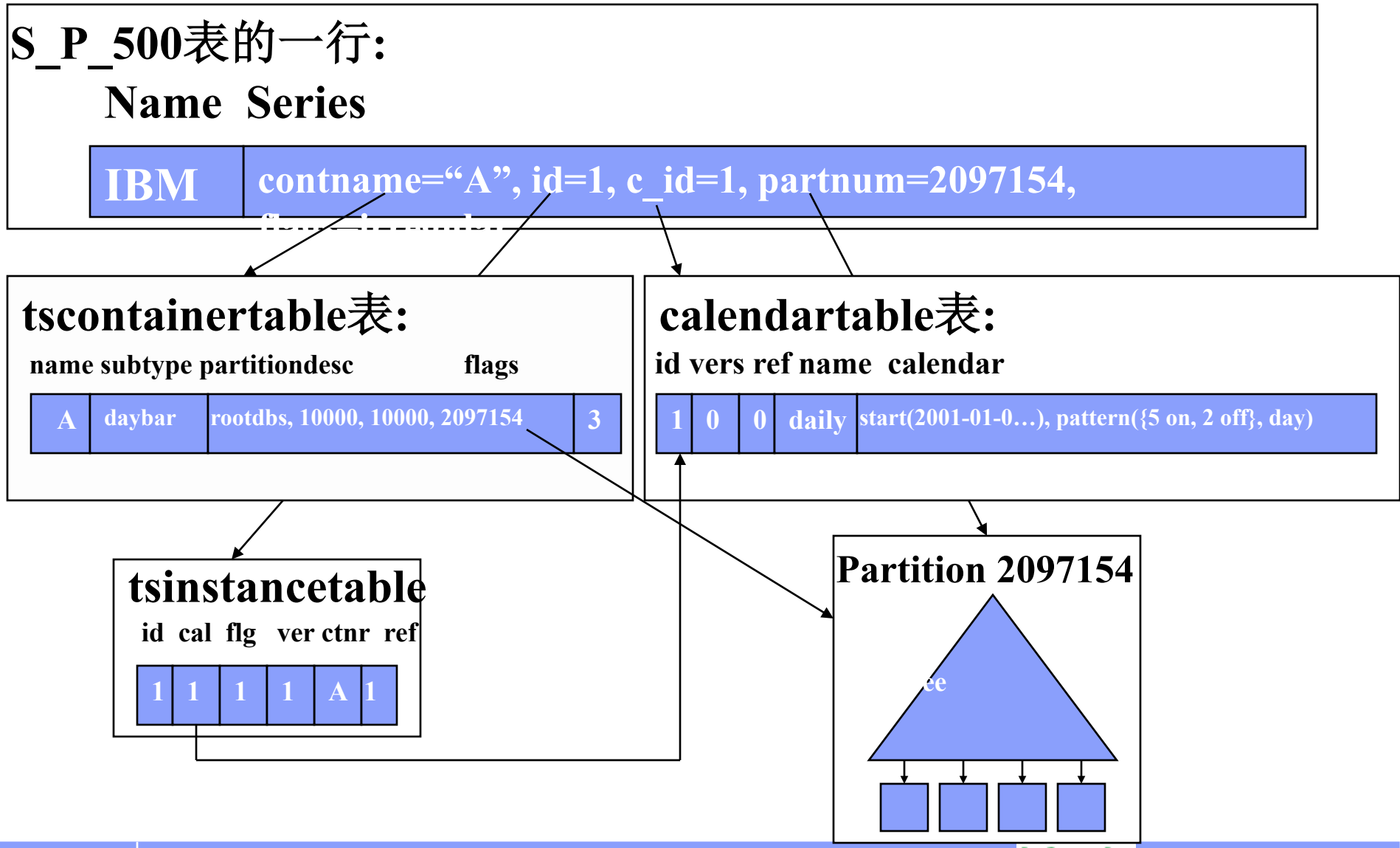
**S_P_500，美國股票價格**

**(lvarchar)　　timeseries(daybar)**

| 股票 | 股票價格儲存時間序列型 |
|---|---|
| IBM | (2010-01-01--00:00, value1), …. |
| ORCL | (2010-01-01--00:00, value1), …. |
| ATT | (2010-01-01--00:00, value1), …. |
| JPM | (2010-01-01--00:00, value1), …. |
| BOA | (2010-01-01--00:00, value1), …. |
| CSCO | (2010-01-01--00:00, value1), …. |
| AAPL | (2010-01-01--00:00, value1), …. |
| GOOG | (2010-01-01--00:00, value1), …. |
| 。 。 。 | 。 。 。 。 |



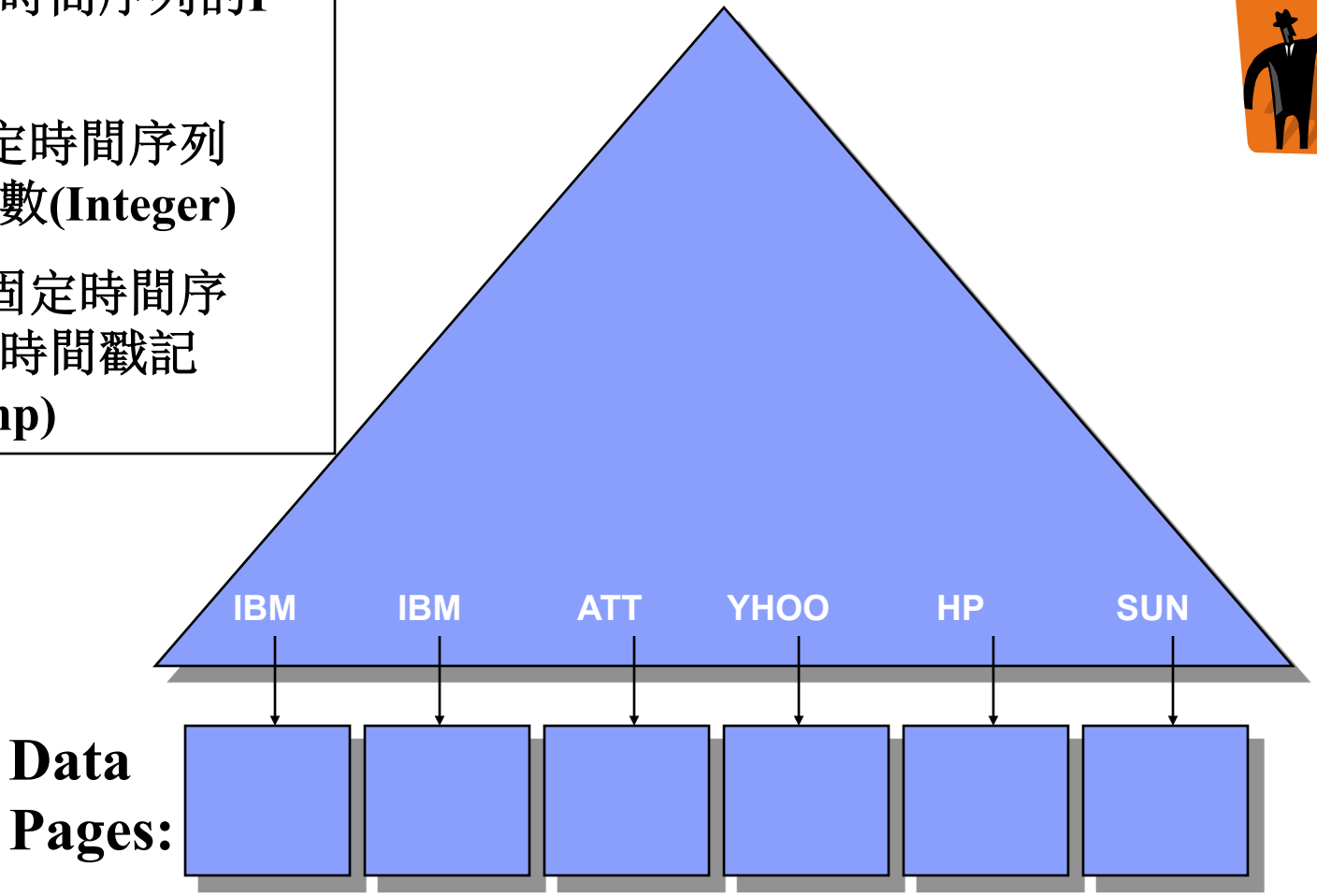
# 挖掘時間序列



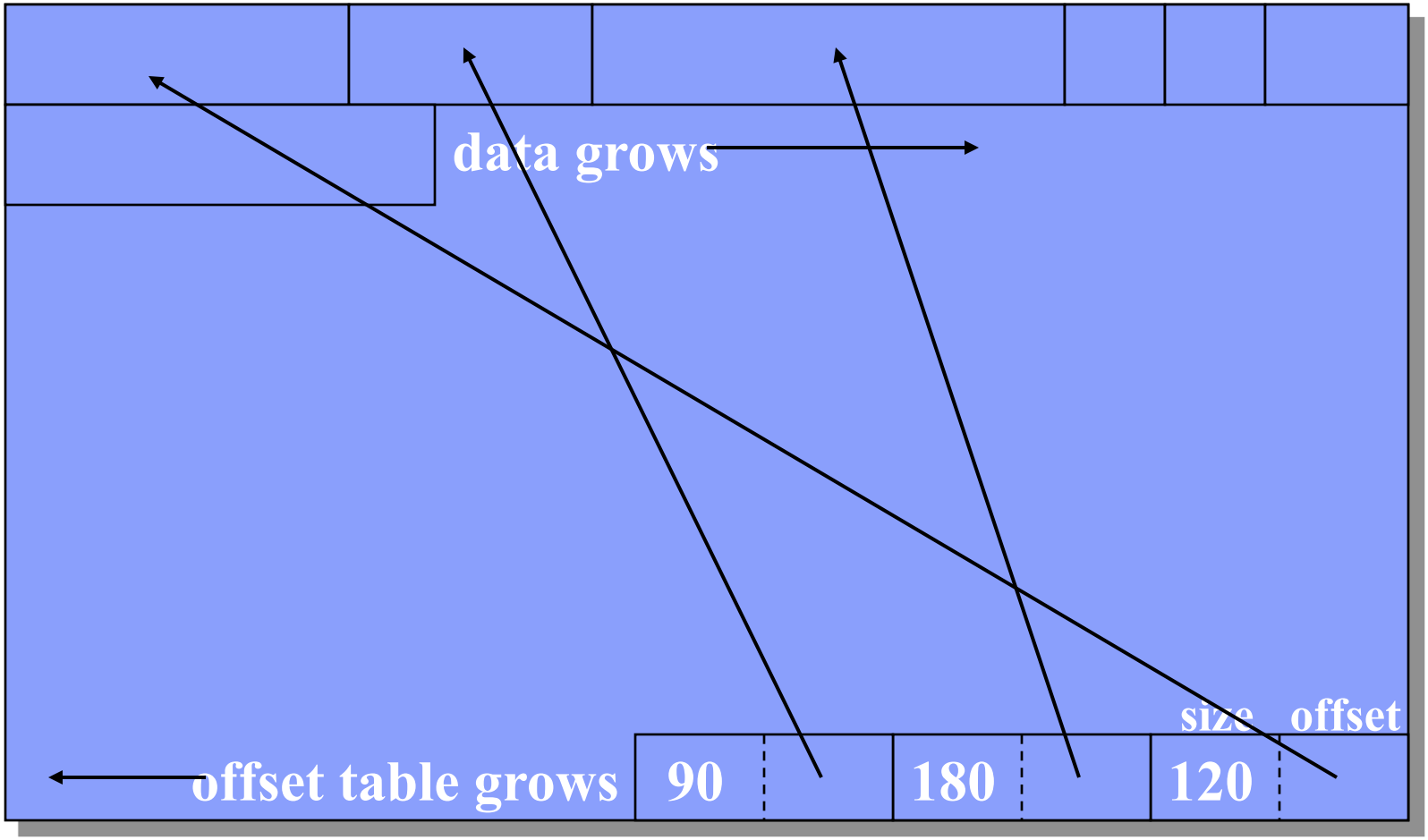
# 容器的結構（Container Layout）

索引鍵是時間序列的ID加上

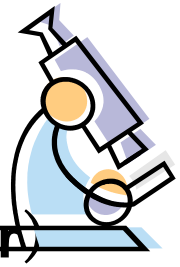
- 對於固定時間序列是一個整數(Integer)
- 對於非固定時間序列是一個時間戳記(timestamp)



# 容器內的Page的結構



# 容器內的行的結構（Row Structure in a Container）



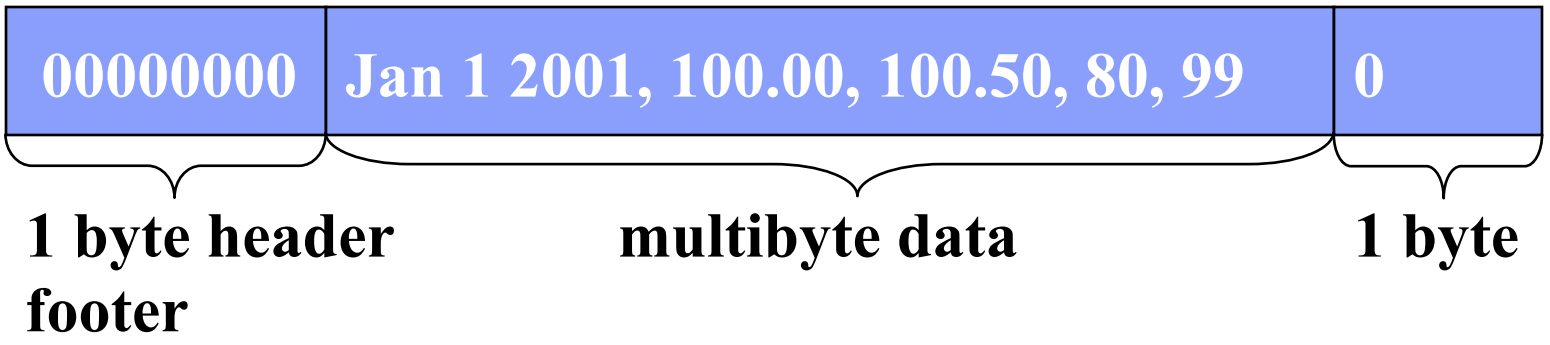
- 時間序列行包含頭（**header),**數據（**data**） 和頁腳（ **footer**）
- 頭長度是**8**位元的整數倍，每“位元”對應行中的每個列
  - 如果列是“null”則相應的“bit”是1，否則是
  - 這樣導致很大的壓縮!
- 資料部分包含來自行內的資料
- 頁腳是一個位元組的長度，它有一個標識行是否為空或隱藏的標誌
  - Bit 2 = 0表示 行是空
  - Bit 3 = 1表示行是隱藏的

# 行，繼續...

- 如果行類被這樣聲明

  ***create row type daybar(tstamp datetime year to fraction (5), bid float, ask float, bidsize int, asksize int);***

- 那麼定義如下的一個沒有空列並且不是隱藏的非固定時間序列:



# 使用時間序列創建應用程式

- 有以下幾個應用介面:
  - SQL
  - VTI
  - SPL
  - Java
  - C-API
- 它是一個工具包!
  - 允許用戶創建自己的分析

# TimeSeries SQL介面

- 通常可以從**SQL**使用者定義常式（**UDR**）存取時間序列數據:
  - Clip() –剪輯一段範圍的時間序列資料並返回
  - LastElem(), FirstElem() –返回時間序列資料的最後/第一個元素
  - Apply() –對一段範圍的時間序列元素應用定義或運算式
  - AggregateBy() –使用彙總函式改變時間序列的間隔
  - SetContainerName() –改變一個容器的名字
  - BulkLoad() –把資料從檔中裝載到時間序列表中

# 時間序列 SQL語句示例

- 取得從**2001**年開始的所有的**IBM**的數據

  *Select Clip(series, '2001-01-01 00:00:00.00000', Current) from S_P_500 where name = 'IBM';*

- 取得**IBM**上一個收盤價

  *Select GetLast(series).close from S_P_500 where name = 'IBM';*

- 查找**IBM**每週的最高價

  *Select AggregateBy('max($high)', 'weeklycal', series, '2001-01-01 00:00:00', Current) from S_P_500 where name = 'IBM';*

# TimeSeries虛表介面（VTI）

- TimeSeries提供另一種工具來降低處理TimeSeries資料的複雜性 這就是用於構建虛表的常式。該功能建立在Informix提供的虛表介面（VTI）之上。虛表使我們可以像“常規的” 表那樣處理TimeSeries數據。它看上去像一個視圖(View)，並且底層資料不存在重複。虛表還可以用於裝載TimeSeries數據。
- 虛表支持SELECT和INSERT語句，但是不支持UPDATE或DELETE語句。但是，如果資料已經存在，那麼INSERTS相當於是更新語句。因此，只有DELETE是受限制的。另一個限制是，不能在TimeSeries虛表上創建索引。清單11展示了如何為棒球例子構建虛表

# 時間序列 VTI介面

- 使時間序列資料看起來像標準的關聯式資料
  - 對於不能處理時間序列的應用程式是有價值的
  - 當應用程式使用ODBC連接時
  - 使用VTI有10%左右的性能代價
- 約束
  - 一個VTI表只能表示基表的一個列
  - 不允許第二個索引
- 創建**VTI**的**SQL**語句:

*execute procedure tscreatevirtualtab('S_P_500_VT', 'S_P_500');*

# VTI介面:繼續

**TimeSeries Table**

**S_P_500**

| 股票 | 股票價格儲存時間序列型 |
|---|---|
| IBM | (2010-01-03--09:30, value1), …. |
| ORCL | (2010-01-03--09:30, value1), …. |
| ATT | (2010-01-03--09:30, value1), …. |
| JPM | (2010-01-03--09:30, value1), …. |
| BOA | (2010-01-03--09:30, value1), …. |
| CSCO | ((2010-01-03--09:30, value1), …. |
| AAPL | (2010-01-03--09:30, value1), …. |
| GOOG | ((2010-01-03--09:30, value1), …. |
| 。 。 。 | 。 。 。 。 |

**TimeSeries Virtual Table**

**S_P_500_VT**

| 股票 | 日期 | 價格 |
|---|---|---|
| IBM | 2010-01-03--09:30 | value1 |
| IBM | 2010-01-03--09:30 | value2 |
| IBM | 2010-01-03--09:30 | value3 |
| IBM | 2010-01-03--09:30 | value4 |
| … | …. | …. |
| ORCL | 2010-01-03--09:30 | value1 |
| ORCL | 2010-01-03--09:30 | value2 |
| ORCL | 2010-01-03--09:30 | value3 |
| … | …. | … |

# 存儲過程(SPL)示例

```
--示例計數非空元素的時間序列
-- Count non-null elements in a time series
--
create function spl_test(arg lvarchar) returning integer
    define var daybar;
    define cnt integer;

    let cnt = 0;
    foreach execute function
            transpose((select series from S_P_500 where name = arg))
       into var
    let cnt = cnt + 1;
    end foreach
    return cnt;
end function;
```

# 時間序列 C-API介面

- 分為用戶端和伺服器端
- 把時間序列處理為表 **(sort of)**
  - 打開（open）和關閉（open）時間序列的函數
  - 在兩個時間戳記內掃描時間序列的函數
  - 創建時間序列的函數
  - retrieve, insert, delete, update函數
- 另外定義了**70**個函數

# C-API示例

```
#include "tseries.h”
/* Example of counting non null elements */
mi_integer
nelems(ts_timeseries *ts, MI_FPARAM *fParam)
{
        ts_tsdesc               *tsdesc;
        ts_tscan                *tsscan;
        ts_tselem                 elem;
        mi_integer                cnt;
        mi_integer                ret;
        MI_CONNECTION *conn;
        conn = mi_open(NULL, NULL, NULL);
        /* open the source timeseries */
        tsdesc = ts_open(conn, ts, mi_fp_argtype(fParam, 0), 0);
        /* scan the entire timeseries */
        cnt = 0;
        tsscan = ts_begin_scan(tsdesc, 0, NULL, NULL);
        while ((ret = ts_next(tsscan, &elem)) != TS_SCAN_EOS)
    if (ret != TS_SCAN_NULL)    /* don't count nulls */
          cnt++;    /* clean up */
        ts_end_scan(tsscan);
        ts_close(tsdesc);
        return(cnt);
}
```

# Batch Loader

# 時間序列模型使用Bulkload來載入數據

在Informix11.7中，Bulkload被設計來載入資料。在bulkload中，一個電錶的資料通過一個檔被載入進資料庫。

```
update dc_table set data =bulkload(data, 'data/input.2002019984', 0) where cp_no = '2002019984';
update dc_table set data = bulkload (data, 'data/input.2002019985', 0) where cp_no = '2002019985';
update dc_table set data = bulkload (data, 'data/input.2002019986', 0) where cp_no = '2002019986';
update dc_table set data = bulkload (data, 'data/input.2002019987', 0) where cp_no = '2002019987';
update dc_table set data = bulkload (data, 'data/input.2002019988', 0) where cp_no = '2002019988';
update dc_table set data = bulkload (data, 'data/input.2002019989', 0) where cp_no = '2002019989';
update dc_table set data = bulkload (data, 'data/input.2002019990', 0) where cp_no = '2002019990';
update dc_table set data = bulkload (data, 'data/input.2002019991', 0) where cp_no = '2002019991';
update dc_table set data = bulkload (data, 'data/input.2002019992', 0) where cp_no = '2002019992';
update dc_table set data = bulkload (data, 'data/input.2002019993', 0) where cp_no = '2002019993';
update dc_table set data = bulkload (data, 'data/input.2002019994', 0) where cp_no = '2002019994';
update dc_table set data = bulkload (data, 'data/input.2002019995', 0) where cp_no = '2002019995';
update dc_table set data = bulkload (data, 'data/input.2002019996', 0) where cp_no = '2002019996';
update dc_table set data = bulkload (data, 'data/input.2002019997', 0) where cp_no = '2002019997';
update dc_table set data = bulkload (data, 'data/input.2002019998', 0) where cp_no = '2002019998';
update dc_table set data = bulkload (data, 'data/input.2002019999', 0) where cp_no = '2002019999';
update dc_table set data = bulkload (data, 'data/input.2002020000', 0) where cp_no = '2002020000';
```

# 增量載入(Incremental loading)

## 關係模型

**load from 1d.unl insert into rt_dc_d_meter;**

sample data of 1d.unl

2001110000|1|2010-01-02 00:00:00|1|1|01/02/09|35411|01/02/09|111.0|111.0|01/02/09|111.0|

2001110001|1|2010-01-02 00:00:00|1|1|01/02/09|35411|01/02/09|111.0|111.0|01/02/09|111.0|

2001110002|1|2010-01-02 00:00:00|1|1|01/02/09|35411|01/02/09|111.0|111.0|01/02/09|111.0|

2001110003|1|2010-01-02 00:00:00|1|1|01/02/09|35411|01/02/09|111.0|111.0|01/02/09|111.0|

2001110004|1|2010-01-02 00:00:00|1|1|01/02/09|35411|01/02/09|111.0|111.0|01/02/09|111.0|

## 時間序列模型

並行載入

```
for org in 11 12 13 14 15 16 17 18 19 20
do
 cat gd$org.unl | \
   ./../TSCLoader "tsdb" 'dc_table' 'cp_no' 'data' 'dc_type' log > log/gd-${org}.log &
done
```

sample data of 1d.unl

2001110000|2010-01-05|00:00:00|1,2,2010-01-20,2010-11-13,12.3,4.5,,6.7,

2001110000|2010-01-05|01:00:00|1,2,2010-01-20,2010-11-13,12.3,4.5,,6.7,

2001110000|2010-01-05|02:00:00|1,2,2010-01-20,2010-11-13,12.3,4.5,,6.7,


***Kevin Brown (kbrown3@us.ibm.com)***


http://www.ibm.com/informix

***Kevin Brown: kbrown3@us.ibm.com***