第三章 研究方法與步驟

第一節 研究方法

由「類似研究及相關文獻資料蒐集」及「現行舊系統的研究」，整理及歸納出影響資料庫效能的因素，本論文將提出一套建置有效率及穩定的超大型資料倉儲之方法，進行實作及效能驗證，藉以證明所設計的方法，確實可以建置一個有效率及穩定的超大型資料倉儲。詳述如下：

1. 藉由「類似研究及相關文獻資料蒐集」及「現行舊系統的研究」，整理及歸納出影響資料庫效能的因素。

2. 提出一套建置有效率及穩定的超大型資料倉儲之方法，利用中華電信固網通聯當作實驗標的，進行實作及效能驗證。

3. 與現行系統比較，藉以證明所設計的方法，確實可以建置一個有效率及穩定的超大型資料倉儲。

第二節 研究步驟

本研究的進行步驟主要分成背景研究階段、需求分析階段、系統設計與建置階段、雙軌測試階段與論文寫作階段。背景研究階段主要以文獻的研讀與現存系統的觀摩為主，透過資料的研讀來設計及修正本研究的方法。需求分析階段主要以現存系統程式詳讀及訪談為主，本研究透過與現有系統開發人員詳談，並盡力蒐集固網通聯相關資訊。在系統設計與建置的階段，利用前一階段需求分析的結果及本研究所設計的方法，建置出一個符合效能及需求的資料倉儲，並進一步進行系統資料的載入與新舊系統資料比對及更正。在雙軌測試階段，蒐集新舊系統效能資訊及進行系統效能調校，同時，驗證本研究所提方法。最後，整合各階段的研究資料進行論文的寫作。本研究之流程如圖 3.2.1 所示。

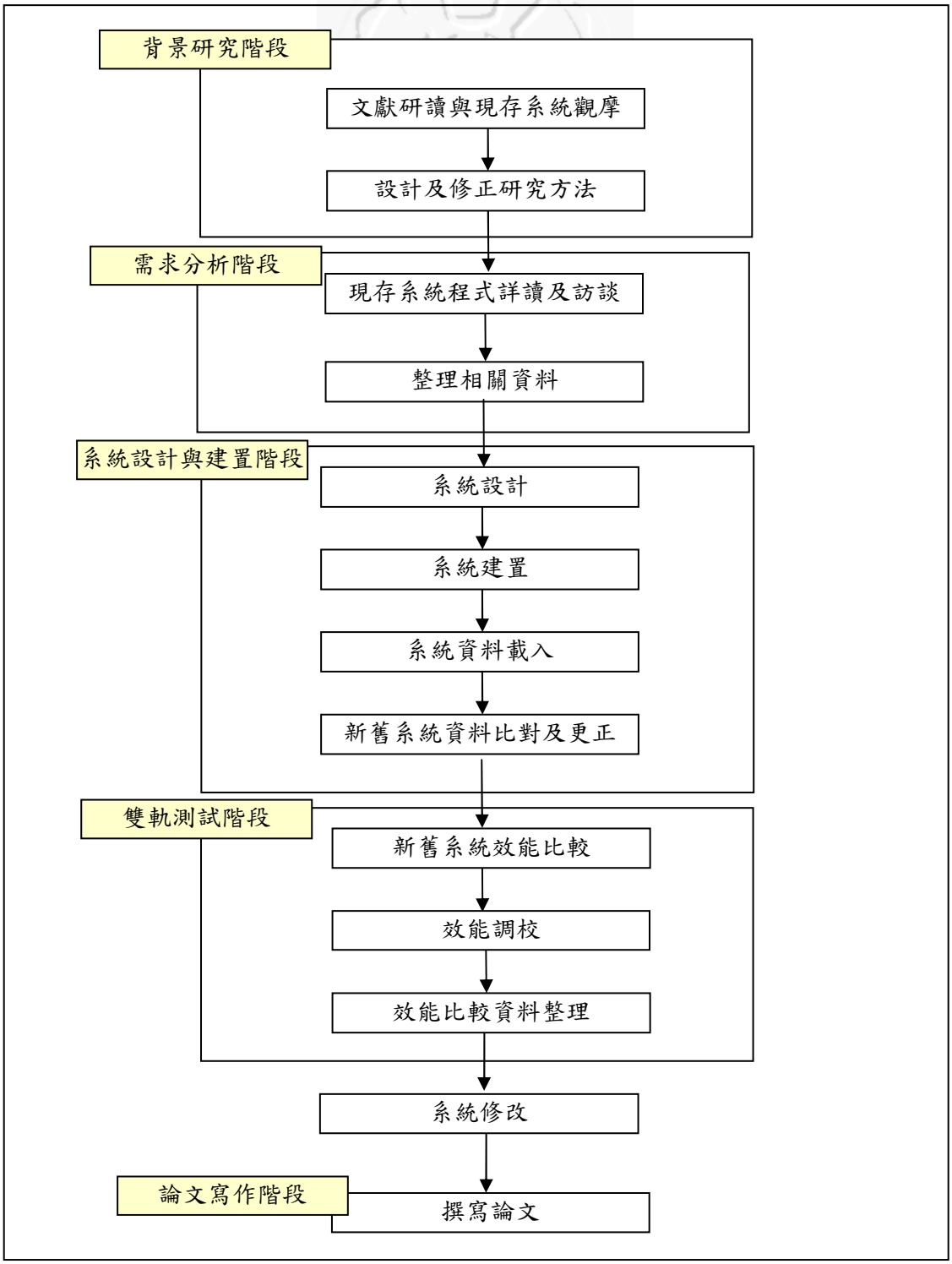
背景研究階段
文獻研讀與現存系統觀摩
設計及修正研究方法
需求分析階段
現存系統程式詳讀及訪談
整理相關資料
系統設計與建置階段
系統設計
系統建置
系統資料載入
新舊系統資料比對及更正
雙軌測試階段
新舊系統效能比較
效能調校
效能比較資料整理
系統修改
論文寫作階段
撰寫論文

圖 3.2.1 研究流程圖

第三節 方法設計

資料庫效能好壞可由新增、異動、刪除、及查詢四方面來決定；達到好的效能方法可能有好幾種，但其所需耗費的成本代價有差異。本研究就效能及成本考量，設計出一套方法，在有限資源之下，有效地發揮資料倉儲的作用。

由於資料倉儲主要在提供決策分析所需資料，亦即首重查詢效率；然而，資料正確性也是相當重要。如果新增、異動之資料無法迅速萃取匯入，勢必無法提供最新資訊，快速查詢固然重要但若所得是過時的資料，查詢再有效率也是枉然。故本研究兼顧資料有效地更新及擷取正確資訊兩方面，提出以下之方法，分項敘述如下：

◆ 資料庫架構設計 － 以集中式取代分散式

就資料庫架構設計而言，資料庫由 32 台伺服器彙整合併(分散式➔集中)，對資料提供的完整性及便利性，有顯著改善。或許有人會對完整性提出質疑：分別對 32 台下 Query 一樣可以達到。但其處理成本就非常昂貴，因很難用簡單資料庫的 SQL 指令去完成任何一項需求;必須先確定 32 台伺服器資料均萃取載入完畢，再一一擷取並確認查詢動作完成，而後將資料一一收集及整理，方能完成一項在集中式資料庫極簡單之查詢工作。所以，分散式資料庫使用者很難做到不用仰賴系統開發人員獨力完成隨意的查詢(ad hoc query)功能，因此，分散式資料庫先天上效率遠不及集中式。

再就成本考量，分散式資料庫，所有需求仰賴系統開發人員，需開發大量應用程式(包括隨意查詢－可能全部再用到的機會很少)。而以程式處理 32 台資料庫的資料時，其異常控制及處理邏輯也遠較集中式複雜許多，人力維護成本大增。就設備而言，分散式備援及管理的工作更複雜，或許設備單價較低，但 32 台資料庫伺服器加總起來－包括管理人力、備援及復原機制難以建立(只能重灌、重 Load)、任何設備異常即影響到資訊提供等考量，相對來講，其代價是較集中式昂貴的。

◆ 克服集中式產生之問題

1. 資料倉儲監控及平行處理的設計－設計一些「記錄資料情況的資料」(簡稱

Metadata）

- 平行處理的設計

  此項設計係起因於由分散式改集中式，為有效提升資料載入速度且避免相同資料重複載入造成異常而產生 — 故於 Metadata 設計 Mutex（全名 **MUT**ual **EX**clusion Object）資訊，各 Process 執行時，遇 Critical Region 必需向系統取得相關 Mutex 方得以執行。之後，各實體視域內容之更新，亦均遵循此模式運作。Mutex 的設計有多種方式，一般多採用作業系統的共享記憶體(Shared Memory)，此處是使用資料記錄，利用資料庫 Primary Key 不可重複特性。既可確保資料正確性及避免同時啟動造成異常，且有助於清楚掌握系統目前運作情形。

- 資料倉儲監控

  除前項 Mutex 資訊外，Metadata 還有錯誤資訊之記錄設計，內容有錯誤發生之 Process、動作、時間、嚴重等級、及通知人員等項目，各 Process 執行時若發生異常一律寫入 Metadata，利於系統監控及進行事後資料補救。

  另外，Metadata 尚需執行紀錄之設計，記載 Process、執行起(迄)時間、筆數等資訊，利於了解執行效能及進行調校。

2. 查詢效能提升的設計

   資料庫查詢效能要好，整理歸納出相關方法如下：

- 各資料表建立適當的索引及 Primary Key(簡稱 PK)

- 對大資料表查詢、Join 資料表需用索引，避免陷入 Full Table Scan

- 所用到的視域(View)，視查詢效能建立實體視域(亦即資料表)

  實體視域在基底關聯表發生異動時，需同步更新，維護成本會增加。故僅在需提供線上查詢資料且效能不佳時，才使用此方法。

- 資料庫連線需設計為可重複使用，適當調整 Application Server 連線儲存池（Connection Pool）最小連線數目，避免一再重新 Connect、

Disconnect，以節省系統資源且提昇查詢的效能。

- 適當調整資料庫開機參數，如 db_writer_processes、parallel_max_servers
- 安排足夠的資料庫表格空間－UNDOTBS、TEMP
- 適當調整常駐程式的數量及休眠時間

3. 異動效能提升的設計

區分為資料萃取載入、資料刪除、實體視域維護及資料庫表格空間，整理歸納出方法如下：

- 資料萃取載入方面，大量資料載入改採用各資料庫系統提供的 Utility，如 Oracle 資料庫的 SQL*Loader、Informix 的 dbload、 Microsoft SQL Server 的 BCP 等。
- 過時不需要的資料刪除（法令規定需保存半年通聯記錄），由於每日資料量極龐大，需採用有效方法來做快速刪除。大量資料異動經常造成資料庫表格空間 UNDOTBS（Oracle 資料 Commit 前使用之暫存空間）滿載而被 Rollback，刪除不掉是極有可能的。利用 Oracle 資料庫 Partition 可以直接 Drop 的特性，設計一套快速有效方法－利用日期做 Partition，即可快速 Drop 某一天資料。
- 實體視域維護方面，當基底關聯表資料量龐大時，應使用 Incremental maintenance，針對異動資料來求算出視域新內容，避免用重新計算方式更新。例如通聯明細每日新增 5 千萬筆，針對這些資料計算出對各實體視域的影響；避免由半年 90 億筆的通聯明細重新計算。為達此，資料萃取載入時，設計一些 Metadata，儲存足以識別各批資料載入實體視域之資訊（如：在電信業固網通聯，可以用通話時間，但自行設計批號效果更好）。Metadata 對各批資料載入時產生唯一可識別的序號，同時在主要基底關聯表之每筆資料記錄增加序號欄位來儲存。Metadata 需能提供各實體視域尚有哪些序號未處理的資訊；甚至所提供資訊能自動產生各實

體視域異動方法，以利加速及簡化實體視域建立與維護。

- 資料庫表格空間－使用大小適中 Raw Device，可加快 I/O 及避免作業系統 cache 而造成異常。

以上本研究所整理及設計的方法，已在中華電信固網通聯系統實作，並經過初步驗證工作 － 比較新舊系統執行效能，根據所獲得數據顯示新系統效能顯著優於舊系統，證實本研究所提出之方法確實能有效地達成新增、異動、刪除、及查詢效能要求。