UNIVERSITÀ DI MODENA E REGGIO EMILIA

Facoltà di Ingegneria - sede di modena
Corso di Laurea Specialistica in Ingegneria Informatica

Tesi di Laurea Specialistica in Informatica

OTTIMIZZAZIONE DI QUERY IN MOMIS

Candidato:
Rodrigue Carlos Nana Mbinkeu

Relatore:
Prof. Sonia Bergamaschi

Correlatore:

Ing. Domenico Beneventano

Anno accademico 2004-2005

Ai miei genitori e amici

"La disumanità del computer sta nel fatto che,

una volta programmato e messo in funzione,
si comporta in maniera perfettamente onesta."
[Isaac Asimov]

Ringraziamenti

Desidero ringraziare anzitutto la Professoressa Sonia Bergamaschi per gli incoraggiamenti e il Professore Domenico Beneventano per la sua grande disponibilità.

Sommario

In un mondo dove le informazioni sono sempre più crescenti e distribuite, i sistemi a mediatori sono una risposta a questa nuova sfida. Le informazioni sono una chiave importante nel processo di decisione sia nelle organizzazioni che per il singolo individuo, avere un'informazione è importante ma averla in tempo breve è importantissimo.

La tesi si colloca nel contesto del sistema MOMIS, sviluppato presso il Dipartimento di Ingegneria dell'Informazione - Università Degli Studi di Modena e Reggio Emilia, ed ha l'obiettivo di ottimizzare le query in MOMIS. Per ottimizzazione si intende la possibilità di ridurre il tempo di risposta delle query cioè il tempo della risposta alla richiesta di un'informazione dev'essere il più breve possibile.

La tesi tratta in particolare l'ottimizzazione algebrica delle query contenenti operatori di outerjoin (full outerjoin, left/right outerjoin). Per risolvere questo problema sono state utilizzate ed estese le tecniche di ottimizzazione presentate in letteratura (ed in particolare da Galindo Legaria): è stato esteso il concetto di *reject null* nel caso specifico di un sistema a mediatore ed è stato introdotto il concetto di partizione degli insiemi.

MOMIS è un sistema di Integrazione Dati basato sulla classica architettura a due livelli wrapper/mediatori; il sistema genera una Vista Virtuale Globale (*Global Virtual View - GVV*) degli schemi locali delle sorgenti da integrare. L'utente può quindi formulare una query sulla GVV e ricevere una risposta unificata come fusione dei risultati provenienti dalle sorgenti locali. MOMIS segue il cosidetto approccio *Global-As-View - GAV*: ogni classe globale della *GVV* è definita da una query sulle classi locali. Tale query è costruita utilizzando l'operatore di full outerjoin. Di conseguenza, ogni query fatta sulla GVV si riconduce ad operazioni di full outerjoin di classi locali. Da questa constatazione è evidente che per ottimizzare le interrogazioni nel sistema MOMIS è fondamentale ottimizzare le operazioni di full outerjoin.

D'altra parte, mentre esistono molti metodi di ottimizzazione per l'operatore di join (sia teorici che implementati in sistemi DBMS commerciali) pochi sono i metodi

di ottimizzazione sviluppati esplicitamente per gli operatore di outerjoin. L'obiettivo della tesi è quindi quello di analizzare tali metodi di ottimizzazione, applicarli ed estenderli nel contesto del sistema MOMIS. Le estensioni dei metodi presentati in letteratura si basano sulla particolarità della query di full outerjoin che definisce una classe globale e sulle ipotesi fatte in un sistema a mediatore come MOMIS.

La tesi è strutturata nel seguente modo:

Nel **primo capitolo** verranno presentati in modo generale i sistemi a mediatori e in particolare verrà descritto il sistema MOMIS.

Nel **secondo capitolo** verranno introdotti tutti gli elementi necessari per definire il problema di ottimizzazione nel contesto del sistema MOMIS; dopo aver definito in generale il problema del Query Processing, daremo la definizione di Sistema di Integrazione, (Classe globale, Mapping table, Join condition e Funzioni di Data Conversion). Verrà affrontato con attenzione il concetto di Full Disjunction, operazione che fa uso dell'operatore di full outerjoin.

Nel **terzo capitolo** si riassumono le principali regole di Semplificazione e Riordino dell'Outerjoin presentate in letteratura, ed in particolare nell'articolo "Outerjoin Simplification and Reordering for Query Optimization" di Galindo-Legaria e Rosenthal: definizione dell'outerjoin, concetto di reject null e proprietà di semplificazione e riordino dell'outerjoin.

Nel **quarto capitolo** faremo l'estensione delle proprietà presentate nel capitolo precende al sistema a mediatore MOMIS. In particolare, introdurremo il concetto di partizione degli insiemi che è alla base delle proprietà di semplificazione dell'outerjoin in MOMIS.

Infine nel **quinto capitolo** affronteremo la riduzione degli schemi delle classi coinvolte in una query. In particolare formalizzeremo un algoritmo per svolgere tale operazione. Inoltre verrà fatto un cenno alla potenza risultante dalla combinazione di tale algoritmo e con le proprietà di semplificazione ottenute nel quarto capitolo.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Integrazione Intelligente delle informazioni

La diversità delle sorgenti di informazioni distribuite e la loro eterogeneità è una delle principali difficoltà riscontrate dagli utenti del Web e della gestione dei dati dei sistemi multi-database oggi. Questa eterogeneità può provenire dal formato opppure della struttura delle sorgenti (sorgenti strutturate: basi di dati relazionali, sorgenti semi-strutturate: documenti XML, oppure non strutturati: testi), dal modo di accesso e della query oppure dell'eterogeneità semantica. Diventa dunque pressoché impossibile da parte di un utente poco esperto, come lo può essere il pubblico sempre più vasto di internet, isolare i dati necessari per una ricerca fruttuosa. In questo ambito, vari progetti stanno studiando nuovi sistemi di supporto alle decisioni, di integrazione di basi di dati eterogenei, i datawarehouse, e di isolamento delle applicazioni e dei dati integrati dalle sorgenti. Integration Information(I^2) si prefigge di combinare tra di loro informazioni provenienti da intere sorgenti o da parti di esse basandosi sulle descrizioni dei dati senza dover ricorrere alla loro duplicazione fisica. Per selezionare le informazioni in modo da ottenere risultati utili è necessaria dunque conoscenza ed intelligenza applicabili alla scelta delle sorgenti e dei dati, alla loro fusione e sintesi. Se a ciò si aggiunge poi l'utilizzo di una Intelligenza Artificiale (IA) che sfrutta le conoscenze acquisite, si arriva non ad una semplice aggregazione di informazioni, ma ad un accrescimento del loro valore, ottenendo nuove informazioni dai dati ricevuti: possiamo allora parlare di Intelligent Integration of Information (I^3).

1.1 L'approccio mediatore

1.1.1 Presentazione generale

L'approccio mediatore [Wiederhold G. (1992)] consiste nel definire una interfaccia tra un agente (umano o software) che sottopone una query ad un insieme di sorgenti potenzialmente pertinenti per rispondere alla query. L'obiettivo è di dare l'impressione di interrogare un sistema centralizzato e omogeneo mentre le sorgenti interrogate sono distribuite, autonome e eterogenee.

Un mediatore comprende uno schema globale, oppure ontologia, il cui ruolo è centrale. E un modello del dominio di applicazione del sistema. L'ontologia fornisce un vocabolario strutturato per il supporto all'espressione delle query. Inoltre, essa stabilisce una connessione tra le differenti sorgenti accessibili. In effetto, in questo approccio, l'integration delle informazioni è fondata sullo sfruttamento delle viste astratte per ottenere una vista globale unificata. La vista globale deve descrivere in modo omogeneo e uniforme i contenuti delle sorgenti nei termini dell'ontologia. Le sorgenti di informazione pertinenti, per rispondere ad una query, sono calcolate per riscrittura della query globale in termini di quelle viste. Il problema consiste a trovare una query che, a seconda della scelta della concezione del mediatore, è equivalente ad una query espressa sulla vista di sorgente. Le risposte alla query sottoposta sono in seguito ottenute valutando le riscritture sulle viste estensionali.

L'approccio mediatore presenta l'interesse di poter costruire un sistema d'interrogazione di sorgenti di dati senza toccare ai dati che rimangono mantenuti nelle loro sorgenti di origine. Così, il mediatore non può valutare directamente le query che gli sono sottoposte perché non contiene i dati, quest'ultimi essendo mantenuti in modo distribuito nelle sorgenti indipendenti. L'interrogazione effettiva delle sorgenti si fa via degli adattatori chiamati Wrapper, che traducono le query riscritte in termini di viste nei linguaggi di interrogazione specifica accettata da ogni sorgente.

L'aiuto fornito dai sistemi di mediazione ricopre diverse forme: scoprire le sorgenti pertinenti dando una query sottoposta, poi aiutare ad accedere a queste sorgenti pernenti, evitando all'utente di interrogare lui stesso ognuna di esse secondo le loro proprie modalità e i loro vocabolari., infine di combinare automaticamante le risposte parziali ottenute dalle diverse sorgenti in modo da dare una risposta globale. Tali sistemi di mediazione offre all'utente una vista uniforme e centralizzata dei dati distribuiti, questa vista corrisponde a una visione più astratta, condensata, qualitativa dei dati e quindi più significante per l'utente. I sistemi di mediazione sono, d'altronde, molto utile , in presenza di dati etereogenei perché danno l'impressione di utilizzare un sistema omogeneo. Fra le differenti grandi categorie d'applicazione di questi sistemi di mediazione, si può citare le applicazioni di ricerca di informazione, quelle di supporto alle decisioni in linea e quelle in maniera più generale di gestione della conoscenza.

A titolo di illustrazione molto semplice di applicazione mediatore, supponiamo che un utente sottopone la query seguente: Quali sono i film di Woody allen a Parigi stasera? Dove? le loro critiche? Supponiamo l'esistenza di due sorgenti di informazione. La prima, Internet Movie Data Base, utilizza un sistema di database relazionale e contiene una lista di film, precisando per ognuno il titolo, gli attori, e il regista. La seconda, Pariscope, può utilizzare dei file XML, contiene, per film, il cinema dove il film può essere visto e per ogni cinema il nome e l'indirizzo. La risposta alla query deve essere costruita interrogando ognuna delle sorgenti e combinando i risultati di interrogazione in modo da offrire all'utente una risposta globale.

Recentemente, nuove applicazioni sono state introdotte nelle aziende: eCRM, Business Intelligence, eERP, eKM, ecc. Queste applicazioni, che si designa a volte con il nome di WebHouse [Kimball R.(2000)] se esse sono utilizzate nel contesto del Web, si appogiano sulla costruzione di magazzino di dati sul Web. Esse si trovano ugualmente confrontate al problema della mediazione poiché esse mettono in opera un processo di acquisizione di dati, a volte in tempo reale, proveniente di sorgenti multiple, distribuite e eterogenee. La realizzazione di applicazione di mediazione intelligente fra gli utenti e le sorgenti d'informazione, accessibili via Web oppure immagazzinato localmente, è necessario. I sistemi di mediazione aiutano l utente a specificare facilmente i dati che egli ricerca,, quest'ultimo avendo l'impressione di utilizzare un sistema unico e omogeneo.

1.1.2 Panoramica dei mediatori esistenti

I differenti sistemi di integrazione di informazione a base di mediatori si distinguono da: una parte, il modo in cui è stabilito la corrispondenza fra lo schema globale e gli schemi delle sorgenti di dati da integrare, d'altra parte i linguaggi utilizzati per modellizzare lo schema globale. gli schemi delle sorgenti dati da integrare e le query degli utenti.

Rispetto al primo punto, si distingue l'approccio Global as Views (GAV) dall' approccio Local as Views (LAV). L'approccio GAV, che proviene dal mondo delle basi di dati federate, consiste a definire lo schema globale in funzione degli schemi delle sorgenti dati da integrare. I sistemi seguente questo approccio sono: HERMES [Subrahmanian V.S et al.(1995)], TSIMMIS [Chawathe S.,Garcia-Molina et al.(1994)], MOMIS. L'approccio LAV è l'approccio duale: essa è adottata nei sistemi seguenti: Razor [Friedman et al.(1997)], Information Manifold [Kirk T. et al.(1995)], OBSERVER [Mena E. et al.(1996).]. I vantaggi dei due approcci sono duali. Secondo l'approccio LAV, è molto facile aggiungere una sorgente d'informazione, ciò non ha effetto sullo schema globale. Però la costruzione della risposta alla query è complessa, invece della costruzione della risposta in un sistema adottando l'approccio GAV che consiste semplicemente a sostituire i predicati dello schema globale della query da le loro definizioni. I sistemi esistenti si differenziano ugualmente dal linguaggio

che utilizzano per esprimere lo schema globale. Si distingue i sistemi fondati sullo schema globale a base di regole (Razor,Hermes,Information Manifold), dei sistemi fondati sullo schema a base di classi (linguaggi orientato object(Tsimmis), Logica descrittiva (Observer,Momis).

1.2 Il Sistema MOMIS

Considerando le problematiche descritte nel paragrafo precedente, nonchè alcuni sistemi preesistenti, si è giunti alla progettazione di un sistema intelligente di integrazione di informazioni da sorgenti di dati strutturati e semi strutturati denominato MOMIS (Mediator EnvirOnment for Multiple Information Sources). Il contributo innovativo di questo progetto rispetto ad altri similari, risiede nell'impiego di un approccio semantico e nell'uso di logiche descrittive per la rappresentazione degli schemi delle sorgenti, elementi che introducono comportamenti intelligenti in grado di rendere semi-automatica la fase di integrazione.Un lavoro approfondito è stato svolto anche riguardo alla fase di query processing, cioè per il processo che dalla query genera automaticamente le sottoquery da inviare alle sorgenti e successivamente ad integrare i risultati. MOMIS nasce all'interno del progetto MURST 40% INTERDATA dalla collaborazione tra i gruppi operativi dell'università di Modena e Reggio Emilia e di quella di Milano.

1.2.1 Problematiche da affrontare

Pur avendo a disposizione gli schemi concettuali delle varie sorgenti, non è certamente un compito banale individuare i concetti comuni ad esse e le relazioni che possono legarli, nè tantomeno è semplice realizzare una loro coerente integrazione. Trascurando le differenze dei sistemi fisici (all'unificazione delle quali dovrebbero provvedere i moduli wrapper) i problemi al livello di mediazione che si è dovuto risolvere (o coi quali si è dovuti scendere a compromessi) sono:

- Problemi ontologici
  Per ontologia si intende, in questo ambito, l'insieme dei termini delle relazioni usate in un dominio, per indicare oggetti e concetti.' In sostanza con ontologia ci si riferisce a quell'insieme di termini che, in un certo dominio applicativo, denotano una particolare conoscenza e fra i quali non esiste ambiguità perché sono condivisi dall'intera comunità di utenti del dominio applicativo stesso.

- Problemi semantici
  Pur ipotizzando che anche sorgenti diverse condividano una visione similare del problema da modelare, e quindi un insieme di concetti comuni, è improbabile che usino la stessa semantica, cioè gli stessi vocaboli e le stesse strutture

dati per rappresentare questi concetti. La causa principale delle differenze semantiche si può identificare nelle diverse concettualizzazioni del mondo reale che persone distinte possono avere, ma non è l'unica. Le differenze nei sistemi di DBMS possono portare all'uso di differenti modelli per la rappresentazione della porzione di mondo in questione: partendo così dalla stessa concettualizzazione, determinate relazioni tra concetti avranno strutture diverse a seconda che siano realizate attraverso un modello relazionale o ad oggetti. L'obiettivo dell'integratore, che è fornire un accesso integrato ad un insieme di sorgenti, si traduce allora nel non facile compito di identificare i concetti comuni all'interno di queste sorgenti e risolvere le differenze semantiche che possono essere presenti tra di loro. Possiamo classificare queste contraddizioni semantiche in tre gruppi principali:

1. eterogeneità tra le classi di oggetti: benché due classi in due differenti sorgenti rappresentino lo stesso concetto nello stesso contesto, possono usare nomi diversi per gli stessi attributi con domini di valori diversi o ancora avere regole differenti su questi valori.
2. eterogeneità tra le strutture delle classi: comprendono le differenze nei criteri di specializzazzione, nelle strutture per realizzare una aggregazione, ed anche le discrepanze schematiche, quando cioè valori di attributi sono invece parte dei metadati in un altro schema(come può essere l'attributo SESSO in uno schema, presente invece nell'altro implicitamente attraverso la divisione della classe PERSONE in MASCHI e FEMMINE);
3. eterogeneità nelle istanze delle classi: ad esempio, l'uso di diverse unità di misura per i domini di un attributo, o la presenza/assenza di valori nulli.

E però possibile sfruttare adeguatamente queste differenze semantiche per arricchire il nostro sistema: analizzando a fondo queste differenze e le loro motivazioni si può arrivare al cosidetto arrichimento semantico, ovvero all'aggiungere esplicitamente ai dati tutte quelle informazioni che erano originariamente presenti solo come metadati negli schemi, dunque in un formato non interrogabile.

1.2.2 L'approccio adottato

MOMIS adotta un approccio di integrazione delle sorgenti semantico e virtuale. Il concetto di semantico è stato illustrato precedentemente. Con virtuale si intende invece che la vista integrata delle sorgenti, rappresentata dallo schema globale, non viene materializzata, ma il sistema si basa sulla decomposizione delle query poste dall'utente e sull'individuazione delle sorgenti da interrogare per generare delle query locali eseguibili su ogni sorgente inerente all'interrogazione; lo schema globale dovrà

inoltre disporre di tutte le informazioni atte al riperimento dei risultati ottenuti localmente.

Le motivazioni che hanno protato all'adozione di un approccio come quello descritto sono varie:

1. La presenza di uno schema globale permette all'utente di formulare qualsiasi interrogazione che sia con esso consistente senza preoccuparsi della rappresentazione strutturale e semantica delle sorgenti che contengono l'informazione cercata;

2. le informazioni semantiche che lo schema globale comprende possono contribuire ad una eventuale ottimizzazione delle interrogazioni;

3. L'adozione di una semantica type as a set per gli schemi permette di controllarne la consistenza facendo riferimento alle loro descrizioni;

4. la vista virtuale rende il sistema estremante flessibile, in grado cioè di supportare frequenti cambiamenti sia nel numero che nel tipo delle sorgenti, ed anche nei loro contenuti(non occore prevedere onerose politiche di allineamento);

Si è deciso di adottare, sia per la rappresentazione degli schemi che per la formulazione delle interrogazioni, un unico modello dei dati basato sul paradigma ad oggetti. Il modello comune dei dati utilizzato nel sistema(ODM_{I^3}) è di alto livello e facilita la comunicazione tra il mediatore ed il wrapper. Per definire questo modello si è cercato di seguire le raccomandazioni relative alla proposta di standardizzazione per i linguaggi di mediazione, nata in ambito I^3: un mediatore deve poter essere in grado di gestire sorgenti dotate di formalismi complessi(ad es. quello ad oggetti) ed altre decisamente più semplici (come i file di strutture), è quindi preferibile l'adozione di un formalismo il più completo possibile. per la descrizione degli schemi si è arrivati a definire il linguaggio ODL_{I^3} che si presenta come estensione del linguaggio standard ODL proposto dal gruppo di standardizzazione ODMG-93.

Per quanto riguarda il linguaggio di interrogazione si è adottato OQL_{I^3} che adotta la sintassi OQL senza discostarsi dallo standard. Questo linguaggio risulta estremamente versatile ed espressivo fornendo la possibilità di sfruttare le informazioni rappresentate nello schema globale. Inoltre si cercato di definire uno standard comune di comunicazione tra i vari moduli MOMIS al fine di rendere ancora più agevole l'ampliamento futuro. Si è deciso di adottare lo standard CORBA per le comunicazioni fra i moduli. CORBA è una tecnologia per l'integrazione, inoltre è ad oggetti ed una modellazione di questo tipo permette di ridurre la complessità di MOMIS: esistono difatti metodologie consolidate per la rappresentazione e progettazione di sistemi ad oggetti (UML), ma soprattutto per utilizzare un oggetto è sufficiente conoscerne l'interfaccia pubblica e questo favorisce il lavoro degli sviluppatori successivi.

1.2.3 L'architettura generale di MOMIS

Momis è stato progettato per fornire un accesso integrato ad informazioni eterogenee memorizzate sia in sorgenti strutturate, come database relazionali, database ad oggetti e semplici file, sia in sorgenti semistrutturate, come le sorgenti descritte in XML.

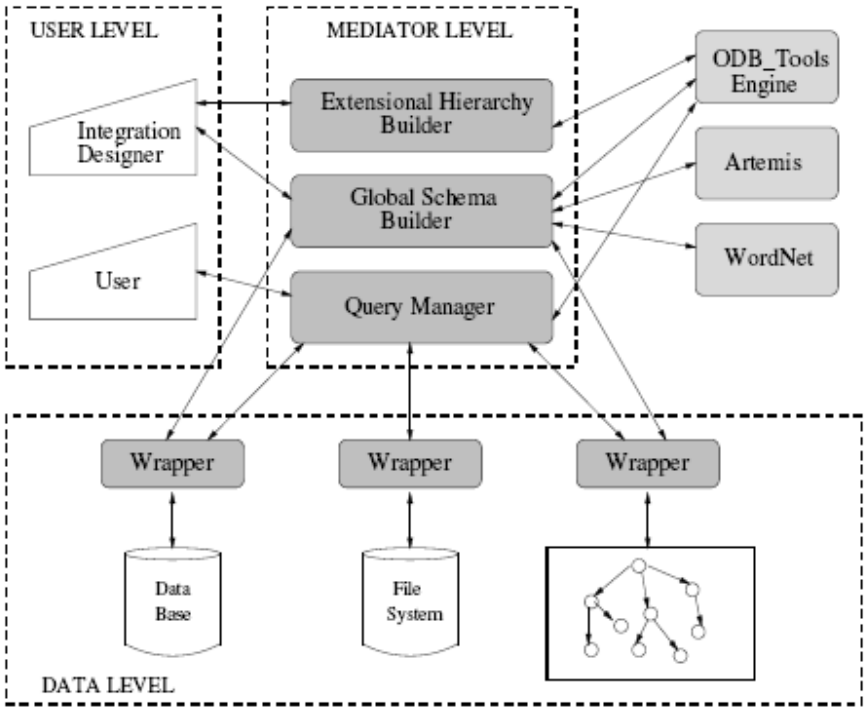

Figura 1.1. Architettura di momis

I componenti del sistema MOMIS sono disposti su tre livelli:

- Livello Dati
  A questo livello si trovano i Wrapper: Posti al di sopra di ciascuna sorgente, sono i moduli che fungono da interfaccia tra il mediatore vero e proprio e le sorgenti locali di dati. Le funzioni da loro svolte sono:

  1. in fase di integrazione forniscono una descrizione delle informazioni contenute nelle sorgenti, utilizzando il linguaggio ODL_{I^3}.

2. in fase di Query Processing, traducono la query ricevuta dal mediatore (espressa in OQL) in una interrogazione comprensibile ed eseguibile dalla sorgente stessa. Inoltre, devono esportare i dati ricevuti in risposta all'interrogazione, presentandoli al mediatore attraverso il modello ODL_{I^3}

   Collegate ai Wrapper ci sono le sorgenti, per questo spesso si parla di quattro livelli.

- Livello mediatore
  Il mediatore è il cuore del sistema ed è composto da tre moduli, ognuno preposto a funzionalità ben precise.

  1. Global schema builder
     La sua funzione principale è quella di generare lo Schema Globale. Il modulo riceve in input le descrizioni degli schemi locali delle sorgenti espressi in ODL_{I^3} e forniti ognuno dal relativo wrapper. A questo punto (utilizzando strumenti di ausilio quali ODB-tools Engine,Wordnet,Artemis) il Global Schema Builder è in grado di costruire la vista virtuale integrata(Global Schema) utilizzando tecniche di clustering e di intelligenza artificiale. In questa fase è prevista anche l'interazione con il progettista il quale , oltre ad inserire le regole di mapping, interviene nei processi che non possono essere svolti automaticamente dal sistema (come ad esempio l'assegnazione dei nomi alle classi globali, la modifica di relazioni lessicali,...).
  2. Extensional Hierarchy Builder
     Il modulo si occupa della gestione della conoscenza estensionale, calcolando strutture dette Base extension e generando la Gerarchia Estensionale.
  3. Query Manager
     E il modulo di gestione delle interrogazioni. In questa fase la singola query posta in OQL_{I^3} dall'utente sullo schema globale(che chiameremo Global Query) sarà suddivisa in più Local Query anch'esse espresse in OQL_{I^3} da inviare alle varie sorgenti, o meglio, come abbiamo visto, ai wrapper predisposti alla loro traduzione. Questa traduzione avviene in maniera automatica da parte del Query Manager utilizzando la conoscenza intensionale e estensionale.

- Livello utente
  L'utente del sistema può interrogare lo schema globale e per lui sarà come interrogare un database tradizionale. La query posta dall'utente sullo schema globale viene trasmessa come input al Query Manager, che interroga le sorgenti

e fornisce all'utente la risposta cercata. Tutte queste operazioni, per l'utente, risultano completamente trasparenti.

1.2.4 Tool di supporto

Per realizzare il processo di integrazione degli schemi il sistema mediatore MOMIS sfrutta anche alcuni tool esterni descritti di seguito:

- ODB-Tools: è uno strumento software svillupato presso il dipartimento di Ingegneria dell'università di Modena e Reggio emilia. Esso si occupa della validazione di schemi e dell'ottimizzazione semantica di interrogazioni rivolte a Basi di Dati orientate agli oggetti (OODB).

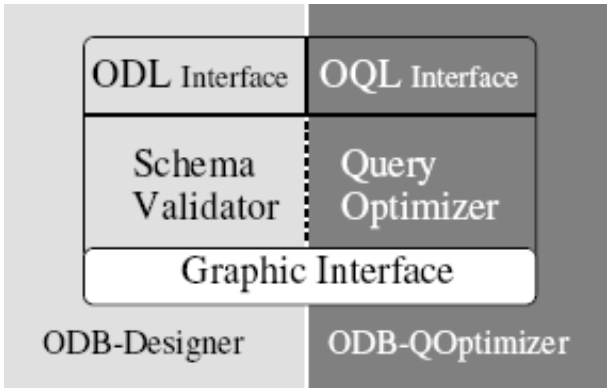

Figura 1.2. Componenti di ODB-Tools

L'architettura di ODB-Tools prevede vari componenti, tra cui:

1. ODB-Designer si occupa della validazione di schemi: si può inserire la descrizione di uno schema di database (in ODL) ed il sistema realizzerà automaticamente la sua validazione e la sua riclassificazione (verifica che non vi siano classi incoerenti e calcola relazioni di specializzazione non esplicitata dallo schema).
2. ODB-Qoptimizer si occupa dell'ottimizzazione semantica delle interrogazioni: se si inserisce una query (in OQL) posta su di un determinato schema, questa viene automaticamente riformulata in una equivalente, ma più efficiente, sfruttando l'espansione semantica ed i vincoli di integrità.

- Wordnet: è un data base lessicale on-line in lingua inglese. Esso è capace di individuare relazioni semantiche fra termini; cioè dato un insieme di termini, Wordnet è in grado di identificare l'insieme di relazioni lessicali che li legano.

- ARTEMIS (Analisys and Reconcialiation Tool Environment for Multiple Information Sources): è uno strumento software sviluppato presso l'università di Milano e Brescia. Riceve in ingresso il thesaurus, cioè l'insieme delle relazioni terminologiche (lessicali e strutturali) precedentemente generate, e sulla base di queste assegna ad ogni classe coinvolta nelle relazioni un coefficiente numerico indicante il suo grado di affinità con le altre classi. Questi coefficienti serviranno per raggruppare le classi locali in modo tale che ogni gruppo (cluster) comprenda solo le classi con coefficienti di affinità simili.

1.2.5 Il linguaggio ODL_{I^3}

Il linguaggio ODL_{I^3} è un linguaggio di definizione attraverso il quale i wrapper comunicano al mediatore (ed in particolare al Global schema Builder) le descrizioni delle sorgenti da loro servite. Punto di partenza per la definizione di questo linguaggio, è stato l'attenersi alle raccomandazioni della proposta di standardizzazione per linguaggi di mediazione. Parallelamente a questa proposta (secondo la quale i diversi sistemi di mediazione avrebbero potuto supportare sorgenti con modelli complessi, come quelli ad oggetti, e sorgenti più semplici, come file di strutture), si è cercato di discostarsi il meno possibile dal linguaggio ODL, a sua volta proposto dal gruppo di standardizzazione ODMG-93. Le caratteristiche peculiari di ODL, al pari di altri linguaggi basati sul paradigma ad oggetti, possono essere così riassunte:

- definizione di tipi-classe e tipi-valore
- distinzione fra intensione ed estensione di una classe di oggetti;
- definizione di attribuiti atomici e collezioni (set, list, bag);
- definizione di relazioni binarie con relazioni inverse;
- dichiarazione della signature dei metodi.

Il linguaggio ODL rappresenta il punto di partenza nel progetto di integrazione; pur essendo, infatti, ben progettato per rappresentare la conoscenza relativa ad un singolo schema ad oggetti, è certamente incompleto se calato in un contesto di integrazione di basi di dati eterogenee. Si è reso pertanto necessario definire un'estensione, denominata OQL_{I^3}, in accordo con le raccomandazioni della proposta di standardizzazione per i linguaggi di mediazione, risultato del lavoro di workshop I^3 svoltosi presso l'università del maryland. Tuttavia, partendo da questa proposta, secondo cui i diversi sistemi di mediazione dovrebbero poter supportare sorgenti con modelli complessi (come quelli ad oggetti) e modelli più semplici (come file di strutture), si è comunque cercato di discostarsi il meno possibile dal linguaggio ODL. I propositi raggiunti dall'estensione di ODL che hanno portato al linguaggio ODL_{I^3} sono i seguenti:

- per ogni classe, è data al wrapper la possibilità di indicare nome e tipo del sorgente di appartenenza;

- per le classi appartenenti ai sorgenti relazionali è possibile definire le chiavi candidate ed eventuali foreign key;

- attraverso l'uso del costrutto union ogni classe può avere più strutture dati alternative, mentre il costrutto optional consente di indicare la natura opzionale diun attribuito. Queste caratteristiche sono in accordo con la strategia utilizzata per la descrizione di dati semistrutturati;

- il linguaggio supporta la definizione di grandezze locali e di grandezze globali;

- il linguaggio supporta la dichiarazione di regole di mapping (o mapping rule fra grandezze globali e grandezze locali);

- è data la possibilità di definire regole di integrità (o if then rule) sia gli schemi locali, sia sullo schema globale;

- il linguaggio supporta la definizione di relazioni terminologiche di sinonimia, ipernimia, iponimia e associazione;

- il linguaggio può essere automaticamente tradotto nella logica descrittiva OLCD usata da ODB-Tools, e quindi utilizzarne le capacità nei controlli di consistenza e nell'ottimizzazione semantica delle interrogazioni.

E da notare come nella fase di progettazione sia compito dei wrapper eliminare le eterogeneità legate al tipo di sorgenti e alla sintassi con cui sono definiti, traducendo la descrizione degli schemi, siano essi relazionali, ad oggetti, semistrutturati,...ecc, in un unico linguaggio descrittivo comune. Utilizzando questo linguaggio, sono fornite dal wrapper al mediatore le descrizioni di tutte le classi da integrare: le desrizioni ricevute rappresentano tutte e sole le classi che una determinata sorgente vuole mettere a disposizione del sistema e quindi rendere interrogabili. Non è detto che lo schema locale ricevuto dal mediatore rappresenti l'intera sorgente: ne descrive il sottoinsieme di informazioni visibili da un utente del mediatore, esterno quindi alla sorgente stessa. Poiché nel sistema MOMIS si è scelto di descrivere le sorgenti basandosi sul paradigma a oggetti, indipendemente dal modello originale adottato da ogni singolo database sorgente, ogni entità verrà descritta dal relativo wrapper utilizzando sempre il concetto di classe. Le sintassi dei linguaggi relativo wrapper utilizzando sempre il concetto di classe.

1.2.6 Il linguaggio OQL_{I^3}

Anche nella definizione del linguaggio di interrogazione OQL_{I^3} si è deciso di adottare la sintassi OQL senza discostarsi dallo standard. Ciò perché nel sistema MOMIS gli elementi di base del piano di esecuzione sono le basic Query e queste, essendo rivolte ad una singola classe globale, non contengono operatori complessi (in particolare join tra classi). Le basic query sono espresse mediante un sottoinsieme del linguaggio OQL, poiché:

- è possibile navigare attraverso aggregazioni e associazioni per ricostruire oggetti complessi ma non sono ammessi join espliciti tra classi;
- non è prevista la presenza di subquery;
- non sono restituite strutture complesse come list, array o struct;
- non sono presenti operatori di ordinamento (order by) o di conversione come:
  1. listtoset: trasforma una lista di elementi in un set, quindi privo di oggetti duplicati.
  2. element: data una collezione di oggetti, ritorna l'elemento in esso contenuto a condizione che sia unico;
  3. atten: trasforma una collezione di collezione ad un solo livello.

Il linguaggio OQL (Object Query Language) è un linguaggio di interrogazione ad oggetti avente una sintassi eccezionalmente chiara e potente simile all'SQL, da cui peraltro deriva: per molti aspetti è un'estensione ad'oggetti di SQL. Questo linguaggio è estremamente versatile ed espressivo perciò se da un lato è necessario uno sforzo maggiore nello sviluppo di moduli per l'interpretazione e gestione delle interrogazioni, dall'altro si ha la possibilità di sfruttare al meglio le informazioni rappresentate nello schema globale. Le principali caratteristiche di questo linguaggio sono:

- OQL è basato sul modello ad oggetti definito da ODMG.
- OQL utilizza una sintassi simile a quella definita per SQL 92. Rispetto a questa presenta delle estensioni finalizzate alla gestione degli aspetti object-oriented, in particolare gli oggetti complessi, l'identità degli oggetti, le espressioni di percorso, il polimorfismo, l'innovazione delle operazioni e il late binding.
- OQL fornisce delle primitive ad alto livello per manipolare insiemi di oggetti, ma non è strettamente legato a questo costrutto. Fornisce infatti anche le primitive per gestire, con la stessa efficienza, altri tipi strutturati come array, liste e strutture.

- OQL non fornisce in modo esplicito operatori per l'aggiornamento della base dati, lasciando questo compito a operazioni opportune che fanno parte delle caratteristiche di ogni oggetto che popola il database. In questo modo si rispetta la semantica propria degli ODBMS che ,per definizione, vengono gestiti attraverso i metodi definiti sugli oggetti.

- OQL permette di accedere agli oggetti in maniera dichiarativa. Questa caratteristica rende più immediata la formulazione dell'interrogazione da parte dell'utente e più semplice ottimizzare le interrogazioni.

Capitolo 2

Query Processing in MOMIS

In questo capitolo verranno introdotti tutti gli elementi necessari per definire il problema di ottimizzazione che vogliamo trattare nella tesi.

2.1 Introduzione al Query Processing in MOMIS

IL sistema mediatore MOMIS è stato sviluppato con l'intento di permettere ad un generico utente di formulare interrogazioni e ricerche e di conseguenza di poter raccogliere informazioni in un contesto eterogeneo e distribuito. Perché ciò sia possibile è necessario risolvere molteplici conflitti intensionali, fornendo una rappresentazione unificata ed omogenea delle diverse sorgenti. In questo capitolo analizzeremo le problematiche inerenti alla gestione della conoscenza intensionale ed all'elaborazione di espressioni di algebra relazionale. Infine presenteremo le operazioni di riscrittura della query globale sulle classi locali e poi l'operazione di data merging e object fusion. Lo scopo di questo capitolo è di presentare i principi che MOMIS usa per dare la risposta ad una query. Per questo MOMIS si appoggia nella prima fase sulla riscrittura della query globale nei termini degli attributi delle sorgenti locali e nella seconda fase sulla full disjunction per ottenere la risposta globale alla query. Data una query sullo schema globale del mediatore, MOMIS individua le classi locali coinvolte nell'interrogazione, suddivide la query in query locali e le invia alle sorgenti corrispondenti. Successivamente recupera le informazioni contenute nelle risposte di ogni query locale e calcola il modo di combinarle per ottenere il risultato globale desiderato. Il nostro scopo è quello di calcolare una istanza dello schema globale paragonabile al concetto di full disjunction presente in letteratura.

Prima di tutto elenchiamo i diversi passi della definizione del query plan e la sua esecuzione:

1. Atomizzazione dei predicati logici contenuti nella query globale(GQ):

   - Individuazione dei predicati logici;
   - Scelta delle regole di riscrittura per i predicati atomici;
   - Riscrittura dei termini atomici;
2. Normalizzazione della GQ;
3. Generazione dell'albero Master rappresentativo;
4. Individuazione delle classi locali coinvolte nella riscrittura;
5. Riscrittura della GQ rispetto alle classi locali;
   - Individuazione dei predicati appartenenti alla GQ e le classi locali per cui tali predicati devono essere riscritti;
   - Generazione dei sotto-alberi per la classe locale individuata;
   - Generazione delle query locali;
6. Mapping logico;
7. Individuazione dei predicati residui e generazione del filtro globale(GF);
8. Recupero delle informazioni e fusione dei risultati (Object fusion);
9. Generazione della Full Disjunction ed esecuzione della GQ;

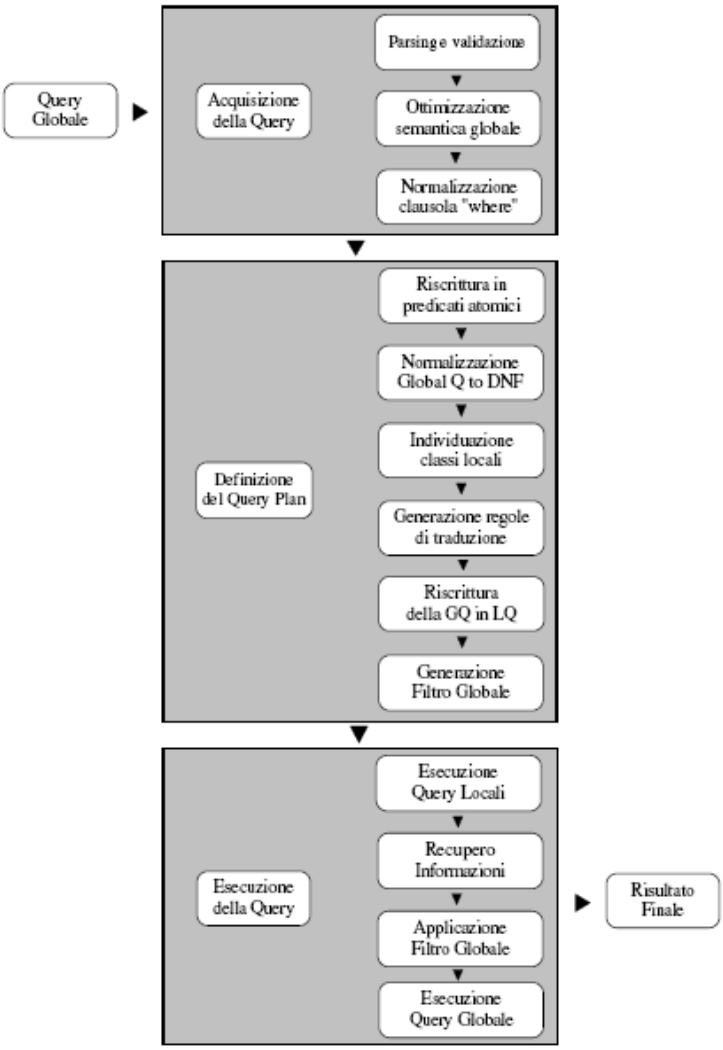

Figura 2.1. Architettura Query Manager

2.2 Il sistema di Integrazione MOMIS

Un sistema di integrazione è definito come $IS = \langle GVV, \mathcal{N}, \mathcal{M} \rangle$ dove:

- (GVV), è la Global Virtual View, uno schema espresso in ODL_{I^3}. I concetti presentati nella tesi si riferiscono, senza perdita di generalità, al caso in cui lo schema sia relazionale.

- Un insieme $\mathcal{N}$ di *sorgenti locali*;

- Un insieme di $\mathcal{M}$ di GAV mapping tra GVV e $\mathcal{N}$, dove ogni asserzione di mapping associa una classe globale di GVV ad una query $q_{\mathcal{N}}$ sugli schemi di un insieme di sorgenti in $\mathcal{N}$.

  Più precisamente, per ogni classe globale $C \in GVV$ definiamo:

  1. un insieme (eventualmente vuoto) di classi locali, indicato con $L(C)$, appartenenti alle sorgenti locali in $\mathcal{N}$.
  2. una query $q_{\mathcal{N}}$ su $L(C)$.

Nella prossima sezione si discute come avviene la definizione di tale query che sostanzialmente calcola l'istanza di una classe globale. Prima di presentare tale definizione occorre introdurre il concetto di *Mapping Table*.

2.2.1 Il concetto di Mapping Table

Il sistema MOMIS genera automaticamente una Mapping Table per ogni classe globale C della GVV, dove le colonne rappresentano le classi locali $L(C)$ appartenenti a C e dove le righe rappresentano gli attributi globali di C. Un elemento MT[GA][L] rappresenta un'insieme di attributi locali di L i quali sono mappati sull'attributo globale GA. Per esempio, la seguente figura mostra la mapping table di Enterprise che raggruppa le classi Company della sorgente L1 e Company della sorgente L2.

2.2.2 Istanza della Classe Globale

Il mediatore, tramite lo schema globale rappresenta la conoscenza disponibile per quanto riguarda gli oggetti del mondo reale istanziati nelle sorgenti locali. Dobbiamo notare subito che tali oggetti del mondo (schema globale) reale non devono essere confusi con quelli (schemi locali) che individuano l'oggetto stesso. Infatti se lo stesso oggetto del mondo reale viene rappresentato in due sorgenti distinte esso deve essere rappresentato a livello del mediatore una sola volta mentre le sue due rappresentazioni nelle due sorgenti locali avranno sicuramente due diversi identificatori. Denotiamo

Enterprise	L1.Company	L2.Company
Address	Address	Address
ContactPerson	–	ContactPerson
Description	Description	AboutUs
EMail	Email	Email
Fax	Fax	–
Name(Join)	CompanyName	Name
Phone	Phone	Tel
Web	HomePage	URL Web

Tabella 2.1. Mapping table di Enterprise

con $\mathcal{O}$ un insieme di oggetti del mondo reale (detti semplicemente oggetti nel seguito) rappresentati al livello del mediatore e introduciamo una funzione ξ, chiamata extension, nel seguente modo: ($\xi : \mathcal{L} \rightarrow 2^{\mathcal{O}}$). Questa funzione associa ogni classe locale L che appartiene all'insieme delle classi locali $\mathcal{L}$ ad un sottoinsieme di classi di $\mathcal{O}$ i cui oggetti sono rappresentati in L. Inoltre con $S(L)$ rappresentiamo lo schema della classe locale L e con A un attributo di $S(L)$. Abbiamo adottato la notazione 'dot' per denotare il valore di un attributo, cioè: se $o \in \xi(L)$ e $A \in S(L)$ allora $o[L].A$ rappresenta il valore dell'attributo A nell'oggetto o appartenente all'estensione $\xi(L)$.

In questa sezione investigheremo su alcune tecniche adottando come riferimento per uno schema e la corrispondente query . Cioè, come detto sopra, ogni classe locale L rappresenta il nome di una relazione con schema $S(L)$ e ogni oggetto è rappresentato in L tramite una funzione di extension (estensione)ξ. In altre parole ogni $o \in \xi(L)$ è rappresentato da una tupla $t_L(o)$ nello schema $S(L)$. In particolare associamo ogni estensione $\xi(L)$ di L con la relazione l di L che ha la seguente estensione $\{t_L(o) | \, o \in \xi(L)\}$. Introduciamo anche la notazione l^G per la relazione con lo schema $S^G(L)$ ottenuta rinominando gli attributi locali di l in quelli globali di G seguendo l'informazione contenuta nella mapping table $\mathcal{M}$ e convertendo opportunamente i dati.

Ad esempio se consideriamo le due classi locali L_1 ed L_2 nel seguente modo:

$$\begin{array}{|l} L_1(firstn,\ lastn,\ year,\ e\_mail) \\ L_2(name,\ e\_mail,\ dept\_code,\ s\_code) \end{array}$$

dopo la fase di integrazione otterremo la classe globale G:

che dunque ha il seguente schema globale: $S(G) = (Name, E\_mail, Section, Year, Dept)$, mentre gli schemi delle classi locali rispetto allo schema globale sono:

$S(L_1) = (Name,\ E\_mail,\ Year)$
$S(L_2) = (Name,\ E\_mail,\ Dept,\ Section)$.

G	name	E_mail	Section	Year	Dept
L1	firstn and lastn	e_mail	–	year	–
L2	name	e_mail	s_code	–	dept_code

Tabella 2.2. Mapping table di G

Per meglio comprendere il concetto diamo anche un esempio con istanze l_1 e l_2:

l_1

firstn	lastn	e_mail	year
Rossi	verde	pv@i.it	2
Ada	Rossi	ra@i.it	1

l_2

name	e_mail	dept_c	s_code
Rossi_Ada	ra@i.it	dept1	12345
Po_Ugo	up@i.it	dept	2345

Tali relazioni dopo le fasi di recupero delle informazioni dalle singole sorgenti locali, di conversione degli attribuiti locali e di modifica dei dati risultano essere con le rispettive notazioni:

l_1^G

Name	E_mail	Year
Rita Verde	pv@i.it	2
Ada Rossi	ra@i.it	1

l_2^G

Name	E_mail	Dept	Section
Ada Rossi	ra@i.it	dept1	12345
Ugo Po	up@i.it	dept1	2345

Poiché sono possibili diversi modi per istanziare lo stesso oggetto, non ci soffermeremo sul problema che riguarda l'integrazione dell'informazione che proviene da sorgenti diverse. Supponiamo che le classi locali soddisfanno la proprietà di omogeneità semantica, cioè tuple che si riferiscono allo stesso oggetto istanziato in diverse classi siano rappresentate nello stesso modo e non vengono mai confuse. Formalmente per ogni coppia di classi locali (L_1, L_2), per ogni $o \in \xi(L_1) \cap \xi(L_2)$ e per ogni $A \in S^G(L_1) \cap S^G(L_2)$ e per ogni $A \in S^G(L_1) \cap S^G(L_2)$ segue che:

$$t^G_{L_1}(o)[A] = t^G_{L_2}(o)[A]$$

oppure:

$$o[L_1].A = o[L_2].A \qquad \begin{array}{l} \forall L_1, L_2, \\ \forall o \in \xi(L_1) \cap \xi(L_2), \\ \forall A \in S^G(L_1) \cap S^G(L_2) \end{array}$$

Ad esempio, considerando le istanze l_1,l_2 e le loro elaborazioni in l_1^G ed l_2^G dell'esempio precedente, possiamo osservare che l'oggetto o presente in entrambe le sorgenti,

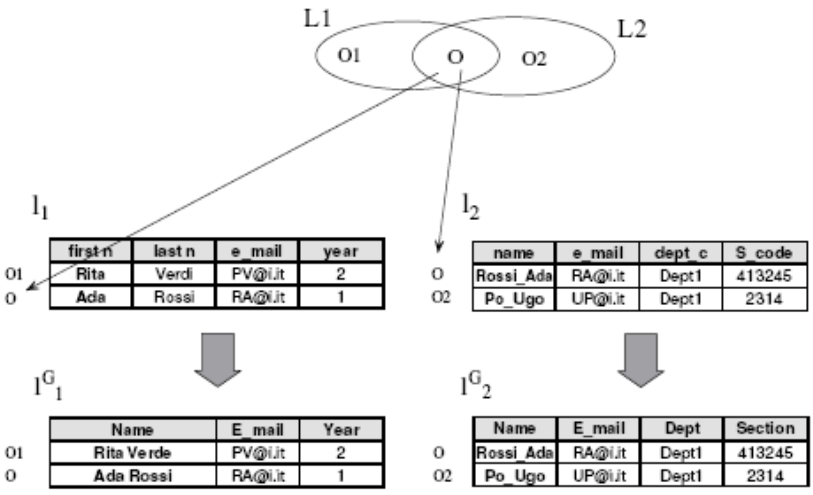

Figura 2.2. Processo di integrazione

per la proprietà di omogeneità semantica ha lo stesso valore per l'attributo E_mail, cioè $l_1^G.E\_mail = l_2^G.E\_mail$.

L'istanza della classe globale G è definita come l'unione minima delle istanze delle classi locali, per cui possiamo definire l'istanza della classe globale come segue:

$$g = \{t_G(o) |\ o \in \mathcal{O}\},$$

le tuple (istanza) della classe globale:

Name	E_mail	Year	Dept	Section
Rita Verde	PV@i.it	2	$\perp$	$\perp$
Ada Rossi	*RA@i.it*	1	*Dept*1	413245
Po Ugo	UP@i.it	$\perp$	Dept1	2314

La seconda tupla è quella che appartiene sia alla sorgente L_1 che alla sorgente L_2.

Sulla base di tali considerazioni, in MOMIS viene utilizzato un metodo per generare automaticamente la query $q_{\mathcal{N}}$ associata ad una classe globale, partendo dalla mapping table ed estendendo l'operatore di *Full Disjunction* che fornisce una "semantica naturale" alle query di fusione dei dati. Nel seguito vengono introdotti gli elementi per la generazione della query.

2.2.3 Funzioni di Data Conversion

Il designer definisce come gli attributi locali vengono mappati sugli attributi globaliGA per mezzo delle *Data Conversion Functions*: per ogni elemento non nullo $MT[GA][L]$ definiamo una *Data Conversion Function*, denotata da $MTF[GA][L]$, il quale rappresenta come gli attributi locali di L sono mappati sugli attributi globali GA. $MTF[GA][L]$ è una funzione che deve essere *eseguita/supportata* dalla sorgente locale della classe L. Per esempio, per sorgenti relazionali, $MTF[GA][L]$ è una SQL value expression; i cui valori di default si mantengono: se $MT[GA][L] = LA$ quindi $MTF[GA][L] = LA$ e, se $MT[GA][L]$ contiene più di un attributo string , allora$MTF[GA][L]$ è la concatenazione di stringhe.
$T(L)$ denota L transformata dalla Data Conversion Function; lo schema di $T(L)$ è composto dagli attributi globali GA tale che $MT[GA][L]$ is non nullo.

Nella presente tesi non ci occuperemo di funzioni di *Data Conversion*, nel senso che le tecniche di ottimizzazione si applicano dopo l'applicazione delle funzioni di Data Conversion; quindi, per semplificare la notazione, nel seguito scriveremo semplicemente L al posto di $T(L)$.

2.2.4 Condizioni di Join

Allo scopo di identificare istanze dello stesso oggetto reale e fonderle si definiscono *Condizioni di Join* tra coppie di classi locali appartenenti alla stessa classe globale. Date due classi locali $L1$ e $L2$ una *Condizione di Join* tra $L1$ e $L2$, denotata con $JC(L1,L2)$, è una espressione su $L1.A_i$ e $L2.A_j$.

Una condizione di join tra L1 e L2 deve essere definita se tra L1 e L2 ci sono delle istanze comuni, ovvero se c'e' intersezione tra le istanze delle due classi. In assenza di vincoli di integrità che dichiarino il contrario, *tra tutte le coppie di classi* appartenenti ad una classe globale, esiste sovrapposizione ed è quindi definita una condizione di join. Ci riferiremo a tale situazione come ipotesi **full connected**.

Un'altra ipotesi è quella che tali condizioni di join siano **consistenti**: intuitivamente, se consideriamo tre classi L1, L2 e L3, il join tra L1 ed L2 (sulla base della condizione JC(L1,L2)) restituisce lo stesso risultato che si ottiene effettuando il join tra L1 e L3 (sulla base della condizione JC(L1,L3)) e tra L2 e L3 (sulla base della condizione JC(L2,L3)).

Questa condizione sembra molto restrittiva ma frequentemente è rispettata da una classe globale; infatti se supponiamo che ogni classe locale L contiene una chiave K e che tutte le join condition sono sulle chiavi, allora si può verificare che siamo in presenza di **join consistenti**.

2.3 Il Concetto di Full Disjunction

Sulla base della discussione fatta nella precedente sezione, Q_N deve essere definita in modo tal che esso contiene una unica tupla risultante dalla fusione di tutte le tuple che rappresentano lo stesso oggetto del mondo reale. Questo problema è relativo a quello dell'elaborazione del natural outer-join di insieme di relazioni in modo da preservare tutte le possibili connessioni tra i fatti(Rajaraman,Ullman, 1996). Tale elaborazione è stata chiamata Full Disjunction (FD) da Galindo Legaria (Galindo-Legaria, 1994)

2.3.1 Full Disjunction

Un problema molto simile al nostro è stato tratta da A. Rajaraman e J.ULman in [27]: quello di fondere insieme la grande quantità di informazione proveniente da molteplici sorgenti eterogenee e distribuite in una struttura che può rendere tale informazione utilizzabile. Gli autori affermano che il problema è legato al fatto di calcolare il natural outerjoin tra tante relazioni in modo da preservare tutte le possibli connessioni tra le relazioni. Il risultato di tale calcolo era stato precedentemente chiamato 'Full Disjunction'(FD, Completa Disgiunzione) da Galindo-Legaria in [6].

Nei lavori appena citati è stato svolto lo studio di quanto la Full Disjunction può essere calcolata tramite una sequenza di outerjoin. La risposta a tale quesito coinvolge il concetto introdotto da Fagin [29] della rappresentazione di schemi di relazioni tramite ipergrafi detto 'ipergrafo γ-aciclico'.

In questa sezione, mostreremo che la risposta globale è ottenuta per mezzo dell'operazione di Full Disjunction delle relazioni locali calcolata rispetto le condizioni di join espresse tramite $\mathcal{F}$. Per poter dare la definizione della Full Disjunction abbiamo bisogno di richiamare velocemente i concetti di schemi e tuple coinvolte e della sussunzione tra tuple.

Si definisce uno schema come un insieme finito di attributi. Una tupla che appartiene ad una classe L si ottiene assegnando dei valori agli attributi di L cioé S(L). Lo schema della tupla t si denota come sschm(t). Alcune tuple possono contenere dei valori nulli e le tuple che contengono solamente tali valori vengono denominate come tuple nulle. Due tuple t_1 e t_2 che appartengono rispettivamente a $S(L_1)$ e $S(L_2)$ possono essere concatenate se gli schemi sono disgiunti oppure se gli assegnamenti coincidono per tutti gli attributi dell'intersezione $S(L_1) \cap S(L_2)$ in tutte e due le tuple. La concatenazione è una tupla t_c tale che $t_c \in S(L_1) \cup S(L_2)$.
Se $t \in S(L)$ possiami ottenere $t^{'} \in S(L^{'})$ con $S(L) \subseteq S(L^{'})$ aggiungendo a t tutti i termini nulli di $S(L) - S(L^{'})$: questo procedimento viene chiamato padding. Un'istanza l sullo schema $S(L)$ è l'insieme fiinito di tuple t_i con schema $S(t_i)$. Denotiamo con S(L) lo schema della relazione L e l la sua istanza. Con questi concetti, un data base è un'insieme di relazioni e l'istanza della data base è l'insieme delle istanze delle relazioni. Diciamo che una tupla t_1 sussume un'altra t_2 se esse hanno lo stesso schema ed inoltre t_2 contiene più valori nulli di t_1 e t_1 coincide con t_2 per tutti gli attributi non nulli. In altre parole t_1 è ottenuta da t_2 sostituendo uno o più valori nulli con valori concreti. Questo vuol dire che tali tuple sussunte rappresentano una ridondanza nelle relazioni che fanno parte della risposta che cerchiamo e dunque devono essere rimosse. La rimozione delle tuple sussunte di un'istanza l(relazione L) ritorna tutte le tuple di l che non siano incluse in nessuna delle altre tuple di l. Possiamo definire che date due relazioni l_1 e l_2 l'unione minima tra di loro è espressa da $l_1 \oplus l_2 := (l_1 \uplus l_1) \downarrow$ e contiene tutte le tuple del risultato dell'outer union rimuovendo le tuple sussunte.

Definizione 2.3.1 *(Full Disjunction) Sia G una classe globale e sia $\mathcal{L} = \{L_1,...,L_n\}$ l'insieme di classi locali corrispondenti. Sia ξ una estensione e siano $\{l_1^G,...,l_n^G\}$ le corrispondenti relazioni le cui tuple non contengono valori nulli. Sia $\mathcal{F}$ una Join Table di G.*
Diremo che una relazione l con lo schema $S(G)$ è la Full Disjunction di $\{l_1^G,...,l_n^G\}$ basata su $\mathcal{F}$ se soddisfa le seguenti condizioni:

1. *Non ci sono ridondanze ed inconsistenze, cioè non esistono delle tuple sussunte.*

2. *Le tuple di l provvengono da pezzi connessi di l_i^ξ: sia t una tupla di l, allora esiste un sottoinsieme di relazioni (istanze) connesse di l_i^ξ tale che t, ristretta ai suoi componenti non nulli, è il risultato dell'operazione di join tra tutte queste relazioni effettuato sulla base della Join Table.*

3. *Tutte le connessioni sono presenti*

   - *siano t_1,...,t_k tuple scelte dalle relazioni distinte $l_{i_1}^\xi$,...,$l_{i_k}^\xi$ rispettivamente, tali che R_{i_1},...,R_{i_k} formano un ipergrafo connesso*

   - *sia t la tupla di l che si mappa (corrisponde) con ognuna dele tuple t_i negli attributi che compaiono in ogni $l_{i_1}^\xi$,...,$l_{i_k}^\xi$ e abbia dei valori nulli negli altri attributi sullo schema di $S(G)$.*

Allora t è sussunta da qualche tupla di l.

Come dimostrato in [29, 27] la Full Disjunction è unica. Il nostro problema fondamentale sarà quello di ottimizzare l'operazione di Full Disjunction e questo è l'argomento principale della tesi.

Ad esempio, riconsiderando le relazioni l_1^G e l_2^G della figura 2.2 abbiamo che la Full disjunction $FD(l_1^G,l_2^G)$ è:

Name	E_mail	Year	Dept	Section
Rita Verde	PV@i.it	2	$\perp$	$\perp$
Ada Rossi	RA@i.it	1	Dept1	413245
Po Ugo	UP@i.it	$\perp$	Dept1	2314

2.3.2 Full Disjunction in MOMIS

Nel nostro contesto: dato una classe globale C composta da $L1$,$L2$,...,Ln, noi consideriamo $FD(L1,L2,...,Ln)$, elaborata sulla base del Join Condition.

Il problema è come calcolare FD. Con due classi, FD corresponde al full outerjoin:

$$FD(L1,L2) = L1 \quad full\,outerjoin \quad L2\ on\ (JC(L1,L2))$$

In (Rajaraman,Ullman, 1996) e stato dimostrato che una sequenza di natural outerjoin produce FD se e solo se l'insieme delle classi locali formano un ipergrafo aciclico. Nel nostro contesto, una classe globale C con più di due classi locali è

un ipergrafo ciclico, per cui non possiamo usare l'algoritmo proposto in (Rajaraman,Ullman,1996).

Il calcolo di FD è compiuta come segue: Noi assumiamo che:(1) ogni classe L contiene una chiave, (2) tutti le join condition sono sulle chiavi (attributi), e (3) tutti gli attributi di join sono mappati sullo stesso insieme di attributi globali, diciamo K. Inoltre, si può dimostrare che : (1) K è la chiave di C, e (2) FD può essere elaborato dalla seguente espressione(chiamato FDExpr):

(L1 full join L2 on JC(L1,L2))

full Join L3 on (JC(L1,L3) OR JC(L2,L3))

...

full join Ln on (JC(L1,Ln) OR JC(L2,Ln) OR ... OR JC(Ln-1,Ln))

Questo è il metodo di calcolo attualmente impiegato in MOMIS; esso verrà indicato nel seguito come approccio o metodo *naive* : l'obiettivo della tei è quello di semplificare e quindi ottimizzare tale espressione.

Dovendo operare delle manipolazioni algebriche, Riportiamo FDExpr in Algebra relazionale introducendo l'operatore $FJ_{\pounds}$, che quindi corrisponderà ad un caso particolare di Full Disjunction (FD).
Nel seguito parleremo solo di $FJ_{\pounds}$.

In Algebra relazionale $FJ_{\pounds}$ è definita induttivamente come segue:

$$FJ_{\{L1\}\cup\{L2\}} = FJ_{\{L1,L2\}} = L1 =\bowtie= L2$$

dove il full join è fatto su JC(L1,L2).

$$FJ_{\{L1,L2\}\cup\{L3\}} = FJ_{\{L1,L2\}} =\bowtie= L3$$

dove il full join è sulla condizione $JC(L1,L3) \vee JC(L2,L3)$.

$$FJ_{\pounds} = FJ_{\{L1,\ldots,Ln-1\}\cup\{Ln\}} = FJ_{\{L1,\ldots,Ln-1\}} =\bowtie= Ln$$

dove il full join è definito da $JC(L1,Ln) \vee JC(L2,Ln) \vee \cdots \vee JC(Ln-1,Ln)$.

Si può verificare che:

$$FJ_{\{L1,L2\}\cup\{L3\}} = FJ_{\{L1,L3\}\cup\{L2\}}$$

ovvero l'ordine seguito per calcolare $FJ_{\pounds}$ è irrilevante.

$$\begin{aligned} FJ_{\{L1,L2\}\cup\{L3\}} &= (L1 =\bowtie= L2) =\bowtie= L3 \\ &= (L1 =\bowtie= L3) =\bowtie= L2 \\ &= FJ_{\{L1,L3\}\cup\{L2\}} \end{aligned}$$

Funzioni di Risoluzione : Omogeneità semantica

La fusione di dati derivanti da sorgenti multiple deve prendere in considerazione il problema di informazioni inconsistenti.

Nel contesto di MOMIS si usano le *Funzioni di Risoluzione* per risolvere i conflitti derivanti da differenti valori di due attributi locali corrispondenti allo stesso attributo globale.

Nel contesto della presente tesi non useremo funzioni di risoluzione ma faremo l'ipotesi di omogeneità semantica introdotta in precedenza. Tale condizione di omogeneità semantica formulata rispetto all'operatore $FJ_{\pounds}$ diventa:

per ogni attributo globale A, per ogni $L_i.A$,$L_j.A \in S(FJ_{\pounds})$, per ogni tupla $t \in FJ$, si ha che

$$t[L_i.A] = t[L_j.A]$$

2.4 Semplificazione Algebrica in MOMIS

In questa sezione verrà introdotto il problema di semplificazione algebrica che vogliamo affrontare in questa tesi nell'ambito del Query Processing in MOMIS.

Consideriamo una query sulla classe global G nella forma:

select $< list >$
from G
where $< condition \quad predicate >$

Siccome per l'istanza della classe G usiamo l'operatore $FJ_{\pounds}$ introdotto prima, la query viene riscritta in algebra relazionale con

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})]$$

dove P è il predicato che esprime la condizione "where" e $\pounds$ sono le classi locali coinvolte nella query che, all'inizio, coincidono con le classi locali relative alla classe globale G.

Nel seguito discuteremo la semplificazione dell'espressione $\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})]$ relativa al termine $X = \sigma_P(FJ_{\pounds})$ e quindi riscriveremo l'espressione iniziale come

$$\Pi_{\mathcal{A}}[\sigma_{F_r}(X^{'})]$$

dove $X^{'}$ è la "riscrittura" di X e F_r è il filtro globale residuo.

Intuitivamente, per "riscrittura" di X si intende una *ottimizzazione/semplificazione* di $FJ_{\pounds}$) basata sulle seguenti trasformazioni

1. **sostituzione dell'operatore di full outerjoin**: alcune operazioni di full outerjoin verranno sostituite con operazioni di join e/o di left/right outerjoin

2. **eliminazione di classi locali** : dall'insieme $\pounds$ verranno escluse alcune classi locali, ovvero il calcolo verrà effettuato su un sottoinsieme di $\pounds$.

3. **push delle proiezioni e delle selezioni** : i termini dell'espressione X', ovvero le relazioni sulle quali verranno fatte le operazioni di join, saranno del tipo $\pi_{AL}(\sigma_{PL}(L))$.

Tale riscrittura verrà trattata nel capitolo "Ottimizzazione dell'Outerjoin in MOMIS", ad eccezione del push delle proiezioni che verrà trattato nel capitolo successivo.

Capitolo 3

Semplificazione e Riordino dell'Outerjoin

In questo capitolo si riassumono le principali regole di Semplificazione e Riordino dell'Outerjoin presentate nell'articolo "Outerjoin Simplification and Reordering for Query Optimization" di Galindo-Legaria e Rosenthal [Gar97].

Tali regole verranno utilizzate ed estese nei capitoli successivi della tesi.

3.1 Definizione dell'operatore di Outerjoin

Il Join di due relazioni (risultati siti locali) restituisce una nuova relazione nella quale ogni tupla è formata dalla giunzione di due tuple, una da ogni relazione originale, tale che i valori degli attributi di join delle due tuple soddisfanno il predicato di join. Di conseguenza, si possono perdere informazioni nel senso che alcune tuple che non soddisfanno il predicato di giunzione non parteciperanno nel risultato del join. Invece, l'outerjoin non perde informazioni. Un outerjoin è una modifica del join che preserva tutte le informazioni provenienti da una oppure entrambe le relazioni [Codd 1979; Lacroix and Pirotte 1979]. Le tuple di entrambi le relazioni che non soddisfanno il predicato di giunzione vengono estese con i valori null per gli altri attributi.

La definizione formale di outerjoin come data in [Date, 1983] è la seguente. Consideriamo R(A, B_1) e S(B_2, C) due relazioni con attributi R.A, $R.B_1$, $S.B_2$ e $S.C$. Definiamo X come il θ-join di R su B_1 con S su B_2, dove θ rappresenta un operatore di confronto:

$$X = J_{R.B_1\ \theta\ S.B_2}(R,S)$$

Dove $J_{R.B_1\ \theta\ S.B_2}(R,S)$ denota il θ-join. Definiamo $R^{'}$ e $S^{'}$ come segue:

$R^{'} = R - X[A,B_1]$

$S' = S - X[B_2,C]$

dove X[A, B_1] e X[B_2, C] sono la proiezione di X su A e B_1 e proiezione di X su B_2 e C, rispettivamente. R' e S' sono quindi le tuple di R e S che non soddisfano il predicato di join. L'outerjoin di R su B_1 con S su B_2, denotato $OJ_{R.B_1\,\theta\,S.B_2}$, è definito come:

$$OJ_{R.B_1\,\theta\,S.B_2} = X \cup (R' \times (\perp,\perp)) \cup ((\perp,\perp) \times S'))$$

Dove $\perp$ denota il valore NULL, e $\times$ denota il prodotto cartesiano. Il left e right outerjoins (oppure one-side outerjoin) di R su B_1 con S su B_2, denotati $LOJ_{R.B_1\,\theta\,S.B_2}$, e $ROJ_{R.B_1\,\theta\,S.B_2}$, sono definiti come:

$LOJ_{R.B_1\,\theta\,S.B_2} = X \cup (R' \times (\perp,\perp))$
$ROJ_{R.B_1\,\theta\,S.B_2} = X \cup ((\perp,\perp) \times S')$

Il left outerjoin preserva le informazioni della relazione di sinistra mentre il right outerjoin preserva le informazioni della relazione di destra.

Se θ è operatore di ugualianza, il θ-join è detto equi-join (questa definizione si applica all'innerjoins e all'outerjoins).

L'esempio seguente [Gar97] mostra un Left Outerjoin che ritorna come risposta alla query i clienti (customers)che vivono in New-York e i loro ordini (orders):

Esempio 3.1.1

```
select all
from CUSTOMER Left Outerjoin ORDERS on
CUSTOMERS.cust=ORDERS.cust
where CUSTOMERS.city='New York'
```

L'operatore di Join nella query può essere eseguito in qualunque ordine dovuto all'associatività e alla comunitavità nell'operazione di Join. Comunque quando entrambi gli operatori di Join e Outerjoin sono presenti nella stessa query, cambiare l'ordine di valutazione è molto complicato. In generale se abbaimo tre relazioni R, S e T allora:
R Left Outerjoin (S Join T) non è equivalente a (R Left Outerjoin S) Join T, oppure neanche a (R Left Outerjoin) Left Outerjoin T. Questo viene mostrato dagli seguenti esempi. Si può assumere che la vista CUSTOMERS_NY ritorna tutti i clienti (customer) che abitano a New-York e la tabella ITEMS mantiene indici che sono nello stock.

Esempio 3.1.2 *Query Originale (QO)*

selecct all
from CUSTOMERS_NY Left Outerjoin (ORDERS Join ITEMS)

Una valutazione diretta di questa query causa la creazione di risultato temporario dell'operazine di join di ORDERS e ITEMS. Molte delle tuple in questo risultato non sono rilevanti perché non sono relative ai clienti (customers) che abitanno in New-York: Quindi vanno eliminate. Un modo per ottimizzare l'elaborazione di questa query è prima effettuare il Left Outerjoin CUSTOMERS_NY e ORDERS in CUST_NY_ORDER.

select all into CUST_NY_ORDER
from CUSTOMERS_NY Left Outerjoin ORDERS

Visto che CUST_NY_ORDERS è il risultato di un outerjoin il quale preserva clienti (customers) che non hanno un'ordine (order) con NULL come valore per tutti gli attributi della tabella ORDERS (ordini). Per questa ragione queste tuple saranno eliminate se CUST_NY_ORDER e ITEMS fanno un Join. Applicazione dell'outerjoin fra CUST_NY_ORDER e ITEMS preservano queste tuple anche se gli items non sono nello stock. In quel caso il risultato non è equivalente alla query originale QO. L'idea è di cercare un operatore che possa computare efficientemente QO ulizzando CUST_NY_ORDER and ITEMS. Questo operatore è specialmente conveniente se gli ordini (orders) emessi dai clienti di New-York sono una porzione della tabella ORDERS (ordini).

3.2 Null-rejection

L'operatore Outerunion $\uplus$ [Cod79] fra due relazioni R_1 e R_2 con l'unione di schemi parzialmente compatibili è definito come segue:

$$R_1 \uplus R_2 = (R_1 \times \{null_{S_2 - S_1}\}) \cup (R_2 \times \{null_{S_1 - S_2}\})$$

Dove $null_A$ è una tupla con NULL come valore in tutti gli attributi di A. Gli esempi seguenti mostrano l'Outerunion fra due relazioni R e S:

A	B	C
a	c	b
d	f	a
c	d	b
c	f	a
c	e	f

$\uplus$

C	D	E
b	g	a
d	a	f

=

A	B	R.C	S.C	D	E	
a	c	b	$\perp$	$\perp$	$\perp$	♣
d	f	a	$\perp$	$\perp$	$\perp$	
c	d	b	$\perp$	$\perp$	$\perp$	♣
c	f	a	$\perp$	$\perp$	$\perp$	
c	e	f	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	b	g	a	♣
$\perp$	$\perp$	$\perp$	d	a	f	

Le tuple con ♣ sono tuple che fanno match nell'operazione di join sull'attributo C. Il Left Outerjoin ($=\bowtie$ oppure $\rightarrow$) e il Full outerjoin ($=\bowtie=$ oppure $\leftrightarrow$) sono :

A	B	R.C	S.C	D	E	
a	c	b	b	g	a	♣
c	d	b	b	g	a	♣
d	f	a	$\perp$	$\perp$	$\perp$	
c	f	a	$\perp$	$\perp$	$\perp$	
c	e	f	$\perp$	$\perp$	$\perp$	

A	B	R.C	S.C	D	E	
a	c	b	b	g	a	♣
c	d	b	b	g	a	♣
d	f	a	$\perp$	$\perp$	$\perp$	
c	f	a	$\perp$	$\perp$	$\perp$	
c	e	f	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	d	a	f	

Il predicato P su un insieme di attributi A è detto **reject nulls**, se esso restituisce **FALSE** oppure **UNKNOWN** quando A è null [Gard97].

Un operatore relazionale (selezione, join, ...) **rejects nulls** su un insieme di attributi A se le tuple con tutti valori nulli di A non influiscono sul risultato dell'operatore

Consideriamo l'esempio del seguente operatore algebrico di selezione:

$$\sigma_{City='NewYork'}(CUSTOMERS)$$

Tutte le tuple con NULL come valore sull'attributo City sono eliminate [Gard97]. Quindi il predicato $City =' NewYork'$ reject nulls su City; anche l'operatore di selezione reject nulls su City.

E' facile verificare che gli operatori di selezione e join rejects nulls se il predicato rejects nulls; il left-outer join $R1 =\bowtie R2$ con predicato p rejects nulls su A se il predicato p rejects null su A e $A \subseteq sch(R2)$. Full outerjoin mai rejects nulls, in quanto restituisce tutte le tuple delle sue relazioni in input.

Secondo la logica dei tre valori (TRUE,FALSE,UNKNOWN) introdotti in SQL2/SQL3 [ANS93; MeS93], il confronto di qualunque valore con NULL ritorna UNKNOWN. Gli operatori logici come NOT e AND ritorna UNKNOWN quando uno dei suoi argomenti is UNKNOWN. Questo significa che il predicato rigetterà (eliminerà) queste tuple. Se l'operatore OR è utilizzato e si intende preservare valori NULL allora l'operatore SQL IsNull può essere utilizzato come mostrato sotto [Gard97]:

```
select
from ORDERS
where ORDERS.priority = 'low' OR IsNull(ORDERS.duedate)
```

3.3 Semplificazione dell'outerjoin

L'idea della semplificazione è basata sui predicati e operatori che reject nulls. Se il risultato di un Left Outerjoin è utilizzato nella stessa query da un operatore che rigettano i valori null delle tuple sul lato destro del Left Outerjoin allora è più efficiente traformare il Left Outerjoin in Join visto che quelle tuple sono eliminate dall'operatore di join come l'esempio seguente:

$$\sigma_{ORDERS.date<1/1/95}(CUSTOMERS =\bowtie ORDERS) =$$
$$\sigma_{ORDERS.date<1/1/95}(CUSTOMERS \bowtie ORDERS)$$

L'esempio assume che i valori null possono apparire solamente nell'attributo date della relazione ORDERS. Per effettuare trasformazioni di questo tipo in [Gard97] sono date le seguenti identità:

$$R_1 \overset{P^1 \wedge P^2}{\rightarrow} R_2 = R_1 \overset{P^1}{\rightarrow} \sigma_{P^2}(R_2), \;\; if \;\; sch(P^2) \subseteq sch(R_2) \qquad (1)$$

$$\sigma_{P^1}(R_1 \overset{P^2}{\rightarrow} R_2) = \sigma_{P^1}(R_1) \overset{P^2}{\rightarrow} R_2, \;\; if \;\; sch(P^1) \subseteq sch(R_1) \qquad (2)$$

Le identità vengono illustrate dai seguenti esempi dove le relazioni R_1 e R_2 sono:

A	B
2	9
7	2
9	3

♣

B	C
0	1
2	3
4	5

♣

In (1) il predicato P^1 è $R_1.B = R_2.B$ e il predicato P^2 è $R_2.C >= 1$. Il risultato del lato sinistro dell'identità (1) è la tabella seguente, il quale è identico al risultato del lato destro dell'identità.

A	R1.B	R2.B	C
7	2	2	3
2	9	⊥	⊥
9	3	⊥	⊥

♣

In (2) il predicato P^1 è $R_1.A > 2$ e il predicato P^2 è $R_1.B = R_2.B$. Il risultato del lato sinistro nell'identità (2) è la tabella seguente, il quale è identico al risultato del lato destro dell'identità.

A	R1.B	R2.B	C
7	2	2	3
9	3	⊥	⊥

♣

Come visto dagli esempi all'inizio della sezione è importante trasformare un Left Outerjoin in un join. Le seguenti identità basate sull'identità (2) sono valide quando il predicato rejects nulls.

$$\sigma_{P_1}(R_1 \overset{P_2}{\rightarrow} R_2) = \sigma_{P_1}(R_1 \overset{P_2}{\bowtie} R_2), \;\; if \;\; P_1 \;\; reject \;\; nulls \;\; su \;\; sch(R_2) \qquad (3)$$

$$\sigma_{P_1}(R_1 \overset{P_2}{\leftrightarrow} R_2) = \sigma_{P_1}(R_1 \overset{P_2}{\leftarrow} R_2), \;\; if \;\; P_1 \;\; reject \;\; nulls \;\; su \;\; sch(R_2) \qquad (4)$$

Per mostrare perché **reject null** è rilevante per la validità dell'identità (2), cambiamo il predicato P_1 nell'esempio precedente per l'identità (2) a $R_1.A > 2$ AND $R_2.B > 1$. Applicando questo predicato produrremo la seguente tabella come un risultato del lato sinistro dell'identità (3):

A	R1.B	R2.B	C
7	2	2	3

♣

Questo è equivalente al risultato del lato destro dell'identità (3). L'identità (4) è verificabile facilmente.

3.3.1 Algoritmo A di semplificazione di Outerjoin con Null rejection

Consideriamo l'albero di una query in figura 3.1. Ogni nodo rappresenta sia un operatore di outerjoin oppure join. Se il predicato P è un operatore che rijects null sull'attributo **A** e **A** è prodotto da un outerjoin nel nodo inferiore allora l'outerjoin del nodo inferiore può essere semplificato usando identità (1-4)per ottenere un one-side outerjoin oppure un join.

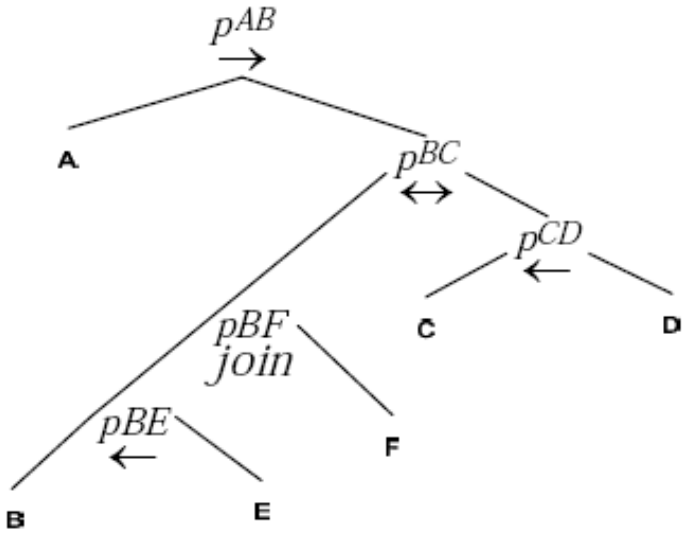

Figura 3.1. Albero Originale

Per esempio se $\overset{P^{AB}}{\rightarrow}$ rigetta nulls su B allora non c'è bisogno di eseguire Full Outerjoin $\overset{P^{BC}}{\leftrightarrow}$, perché tutte le tuple che hanno valori nulli sul B sono rigettati nell'operazione successiva da $\overset{P^{AB}}{\rightarrow}$. Questo implica che $\overset{P^{BC}}{\leftrightarrow}$ può essere trasformato in $\overset{P^{BC}}{\rightarrow}$ in modo da preservare anche tuple che hanno valori nulli su C. Con la stessa logica: visto che valori nulli su B nel right outerjoin $\overset{P^{BE}}{\leftarrow}$ saranno eliminati (rejects null) in $\overset{P^{AB}}{\rightarrow}$ allora esso può essere trasformato in $\overset{P^{BE}}{\bowtie}$. Dopo l'applicazione di queste semplificazioni su l'albero originale della query, otteniamo un albero detto albero semplificato della query mostrato in figura 3.2. In [Gard97] è presentato un algoritmo per effettuare queste semplificazioni.

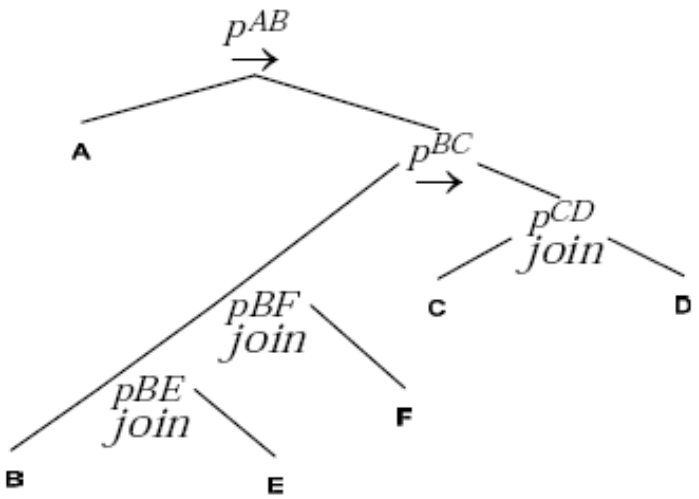

Figura 3.2. Albero Semplificato

3.3.2 Associatività fra Join e Outerjoin

Oltre alle regole di semplificazioni, nel query processing servono regole di associatività per cambiare l'ordine degli operandi. Nel seguito sono riportate alcune di queste regole, prese da [Gard97]:

$$(R_1 \overset{P^{12}}{\bowtie} R_2) \overset{P^{12}\&P^{23}}{\bowtie} R_3 = R_1 \overset{P^{12}\&P^{13}}{\bowtie} (R_2 \overset{P^{23}}{\bowtie} R_3) \tag{5}$$

$$(R_1 \overset{P^{12}}{\bowtie} R_2) \overset{P^{23}}{\rightarrow} R_3 = R_1 \overset{P^{12}}{\bowtie} (R_2 \overset{P^{23}}{\rightarrow} R_3) \tag{6}$$

$$(R_1 \overset{P^{12}}{\rightarrow} R_2) \overset{P^{23}}{\rightarrow} R_3 = R_1 \overset{P^{12}}{\rightarrow} (R_2 \overset{P^{23}}{\rightarrow} R_3), if\ p^{23}\ reject\ null\ on\ sch(R_2) \tag{7}$$

$$(R_1 \overset{P^{12}}{\leftarrow} R_2) \overset{P^{23}}{\rightarrow} R_3 = R_1 \overset{P^{12}}{\leftarrow} (R_2 \overset{P^{23}}{\rightarrow} R_3) \tag{8}$$

$$(R_1 \overset{P^{12}}{\leftrightarrow} R_2) \overset{P^{23}}{\leftrightarrow} R_3 = R_1 \overset{P^{12}}{\leftrightarrow} (R_2 \overset{P^{23}}{\leftrightarrow} R_3), if\ p^{12}\ e\ p^{23}\ reject\ null\ on\ sch(R_2) \tag{9}$$

$$(R_1 \overset{P^{12}}{\leftrightarrow} R_2) \overset{P^{23}}{\rightarrow} R_3 = R_1 \overset{P^{12}}{\leftrightarrow} (R_2 \overset{P^{23}}{\rightarrow} R_3), if\ p^{23}\ reject\ null\ on\ sch(R_2) \tag{10}$$

L'identità (7), verrà utilizzato nella prossima sezione per sviluppare un generalized outerjoin, esso richiede predicati che rigettano (eliminano) null. Il seguente esempio mostra il caso dove (7) non valido quando il predicato accetta nulls.
Relazioni R, S e T sono :

A
1

A	B
2	$\perp$

A
3

Il predicato di left outerjoin fra R e S p^{RS} è $(R.A = S.B)$. Il predicato di left outerjoin fra S e T p^{ST} è $(S.C = T.C \quad or \quad IsNull(S.C))$. Applicando p^{RS} e p^{ST} otteniamo:

$R \overset{P^{RS}}{\rightarrow} S$

A	B	S.C
1	⊥	⊥

$S \overset{P^{ST}}{\rightarrow} T$

B	S.C	T.C
1	⊥	⊥

Applicando l'outerjoin secondo la (7) otteremo due risultati diversi mostrato sotto:

$(R \overset{P^{RS}}{\rightarrow} S) \overset{P^{ST}}{\rightarrow} T$

A	B	S.C	T.C
1	⊥	⊥	3

$\neq$

$R \overset{P^{RS}}{\rightarrow} (S \overset{P^{ST}}{\rightarrow} T)$

A	B	S.C	T.C
1	⊥	⊥	⊥

3.3.3 Generalized Outerjoin (GOJ)

In un'operazione di outerjoin vengono preservati tutti i valori degli attributi degli operandi. Con il Generalized Outerjoin si specifica un insieme di attributi A e si vuole preservare la proiezione su A dei suoi operandi [GaR92].

Generalized outerjoin su p delle relazioni R_1, R_2 preservando attributi $A \subseteq R_1$ è definito come:

$$R_1 \ GOJ[\,p,A]\ R_2 = (R_1 \overset{P}{\bowtie} R_2) \uplus (\pi_A R_1 - \pi_A(R_1 \overset{P}{\bowtie} R_2))$$

Per mostrare come GOJ aiuta nell'elaborazione la Query Originale (QO) menzionata nell'introduzione, un esempio è sviluppato e spiegato sotto.

Sulla base della definizione sopra, le seguenti identità sono state sviluppate:

$$R_1 \overset{P^{12}}{\rightarrow} (R_2 \overset{P^{23}}{\bowtie} R_3) = (R_1 \overset{P^{12}}{\rightarrow} R_2)\ GOJ[\,p^{23},sch(R_1)]\ R_3, \tag{11}$$

$if\ p^{23}\ reject\ null\ on\ sch(R_2)$

$$R_1 \overset{P^{12}}{\leftrightarrow} (R_2 \overset{P^{23}}{\bowtie} R_3) = (R_1 \overset{P^{12}}{\leftrightarrow} R_2)\ GOJ[\,p^{23},sch(R_1)]\ R_3, \tag{12}$$

$if\ p^{23}\ reject\ null\ on\ sch(R_2)$

Esempio di uso di GOJ

Questo esempio mostra nei minimi dettagli tutti i passi nel usare l'operatore GOJ per produrre risultati della query originale CUSTOMERS_NY→(ORDERS ⋈ ITEMS). La relazione CUSTOMERS_NY contiene dati sui clienti vhe vivono in NEW YORK. Per semplicità, assumiamo che ogni ordine (order) ha un item unico. I clienti 1, 2, 3 hanno fatto un ordine per un item valido nello stock. In più il cliente 3 ha fatto un ordine per un item fuori dallo stock. I clienti 4 e 5 hanno fatto nessun ordine. Il cliente 6 ha fatto un ordine per un item fuori dallo stock. La relazione ORDERS contiene gli ordini chiesti da tutti i clienti. Molti di questi ordini sono nello stock. La relazione ITEMS contiene items che sono nello stock. Assumiamo che le relazioni CUSTOMERS_NY, ORDERS, e ITEMS hanno i seguenti contenuti (solamente gli attributi rilevanti sono mostrati):

CustId
1
2
3
4
5
6

CustId	OrderId	ItemId	
1	10	100	
2	20	200	
3	30	300	
3	300	3000	**
6	60	600	**

OrderId	ItemId
10	100
20	200
30	300

Le tuple con ** nella relazione ORDERS e attraverso questo esempio indica l'ordine che è associato ad un item fuori dallo stock.
Applicando il left outerjoin fra CUSTOMERS_NY e ORDERS otteremo:

$$\text{CUSTOMERS\_NY} \overset{P1}{\rightarrow} \text{ORDERS} \qquad (1)$$

CustId	OrderId	Itemid	
1	10	100	
2	20	200	
3	30	300	
3	300	300	**
6	60	600	**
4	⊥	⊥	♠
5	⊥	⊥	♠

Dove $p1$ è $CUSTOMERS\_NY.CustId = ORDERS.CustId$. le tuple con ♠ indica i clienti che non hanno fatto ordine.
Applicando join fra ORDERS e ITEMS otteremo:

$$\text{ORDERS } \overset{P^2}{join} \text{ ITEMS} \tag{2}$$

CustId	ORDERS.OrderId	ORDERS.Itemid	ITEMS.OrderId	ITEMS.ItemId
1	10	100	10	100
2	20	200	20	200
3	30	300	30	300

Dove p^2 è :
ORDERS.OrderId=ITEMS.OrderId AND ORDERS.ItemId=ITEMS.ItemId.
Applicando join fra (1) e (2) otteniamo un risultato che non contiene tuple dei clienti 4 e 5.
Applicando il left outerjoin fra (1) e (2) otteniamo:

$$(CUSTOMERS\_NY \overset{P^1}{\rightarrow} ORDERS) \overset{P^2}{\rightarrow} (ORDERS \overset{P^2}{join} ITEMS) \tag{3}$$

CustId	ORDERS.OrderId	ORDERS.Itemid	ITEMS.OrderId	ITEMS.ItemId	
1	10	100	10	100	
2	20	200	20	200	
3	30	300	30	300	
3	300	3000	$\perp$	$\perp$	**
6	60	600	$\perp$	$\perp$	**
4	$\perp$	$\perp$	$\perp$	$\perp$	♠
5	$\perp$	$\perp$	$\perp$	$\perp$	♠

Così il risultato mantiene ordini 300 e 60, che sono fuori dallo stock. Applicando (11) alla query originale, otteniamo:

$$CUSTOMERS\_NY \rightarrow (ORDERS \ join \ ITEMS) =$$

$$CUSTOMERS\_NY \rightarrow ORDERS \ \ GOJ\,[sch(CUSTOMERS\_NY)] \ ITEMS$$

Il risultato del lato destro dell'equazione applicato una volta sola sulle relazioni utilizzate in questo esempio sarà:

CustId	ORDERS.OrderId	ORDERS.Itemid	ITEMS.OrderId	ITEMS.ItemId
1	10	100	10	100
2	20	200	20	200
3	30	300	30	300
6	⊥	⊥	⊥	⊥
4	⊥	⊥	⊥	⊥
5	⊥	⊥	⊥	⊥

GOJ mantiene i clienti 4 e 5 e rimuove l'ordine 300.

Se assumiamo che ORDERS contiene 1 milione di altri ordini che sono rilevanti per la query originale allora il guadagno ottenuto da GOJ è significante. Questo è perché il risultato intermedio di ORDERS join ITEMS produce sarà vicino ad un milione di tuple [Gard97].

3.4 Conclusione

Le Queries che contengono join e outerjoin possono essere riordinate in modo tale che il costo totale di esecuzione delle queries viene ridotto perché combinazioni meno pesante in termine di costo vengono eseguite per primo rispetto a quelle più pesanti [Gard97].

Le identità algebriche presentate in questo capitolo contribuiscono alla semplificazione di query contenente join, outerjoin e selezione.

L'albero della query può essere semplificato sulla base dell'assunzione che quei predicati e l'operatore di selezione rigettano (eliminano) null. [Gard97]

Il Generalized outerjoin computa efficientemente queries che contengono entrambi join e outerjoin. [Gard97]

È diventato molto importante gestire database come un deposito di risorse e permettere alle applicazioni di accedere a queste risorse in un ambiente distribuito. Un sistema di multidatabase, è un sistema il cui obiettivo è di provedere un accesso unificato e integrato ad una collezione di database esistenti.

Le viste globali sono definite per permettere di ottenere un unica tupla risultante dalla combinazione delle tuple appartenenti alle diverse viste locali.

Molti algoritmi di elaborazione per query distribuite sono stati proposti. Alcuni fanno uso di semijoin per ridurre la quantità di dati da trasferire; alcuni fanno uso di strategia di frammentazione e replicazione per permettere l'elaborazione parallela; alcuni integrano queste due strategie per adattarlo a diversi tipi di reti di computer;

oltre a ciò alcuni applicano informazioni semantiche per velocizzare l'elaborazione della query. In questo capitolo, l'ottimizzazione di query in multidatabase relazionali è stato considerato. Outerjoin è utilizzato per integrare schemi locali in modo da ottenere una vista globale. Sappiamo che per il nostro sistema mediatore MOMIS, otteniamo la vista globale per mezzo della full disjunction che fa uso di un solo operatore algbrico che è il Full Outerjoin. Il nostro obiettivo nei capitoli successivi è di riuscire a trasformare alcuni Full Outerjoin in Left(Right) Join oppure Join. Sulla base della struttura della query e la definizione degli schemi, query modificate sono ottimizzate. Nessuno degli algoritmi citati sopra considerano l'ottimizzazione dell'elaborazione di Outerjoin.

Capitolo 4

Ottimizzazione dell'Outerjoin in MOMIS

Nel sistema di integrazione MOMIS la query associata ad una classe globale (e che quindi ne individua la sua istanza) e' definita sulla base dell'operatore di full outerjoin; e' quindi possibile applicare tutte le tecniche di ottimizzazione (semplificazione e riordino dell'OuterJoin) viste bel capitolo precedente.

L'obiettivo di questo capitolo, che costituisce anche uno degli obiettivi principali dell'intera tesi, e' quello di applicare ed estendere le tecniche di ottimizzazione alla query di outerjoin associata ad una classe globale.

Per estensione si intende in particolare una specializzazione delle tecniche citate al caso specifico, dove viene considerato l'outerjoin non di generiche classi (relazioni) ma di un insieme di classi L_1,L_2,$\ldots L_n$ relative ad una classe globale, per le quali valgono le seguenti ipotesi:

1. **omogeneità semantica** :

   il valore di un attributo globale puo' essere recuperato indifferentemente da una delle classi nelle quali e' mappato;

2. **join consistenti** : nella valutazione dell'outerjoin l'ordine e' irrilevante

3. **full connected**: tra tutte le coppi di classi L_i,L_j esiste una condizione di join

Intuitivamente, grazie all'omogeneità semantica, si potranno effettuare semplificazioni relative alla select list della query, individuando dei casi in cui escludere dall'outerjoin una o più classi. Le altre due ipotesi sono particolarmente utili per cambiare l'ordine degli operandi nell'operazione di outerjoin; nel resto del capitolo verra' formalmente discusso l'uso di tali ipotesi.

I metodi di semplificazione/ottimizzazione proposti nel seguito si basano sul concetto di NULL REJECTION introdotto da Galindo Legaria e discusso nel capitolo precedente. Tale concetto di NULL REJECTION verrà utilizzato ed esteso per considerare le particolarità proprie del sistema a mediatore MOMIS; inoltre verrà introdotto un concetto duale, che chiameremo NOT NULL PROJECTION che riguarda gli attributi select di una query e serve a considerare come valide solo quelle tuple in cui almeno un attributo di select eÕ non nullo, ovvero ad eliminare dal risultato della query quelle tuple in cui tutti i valori delle degli attributi di select sono nulli.

4.1 Estensione delle proprietà elementari

In questa sezione le proprietà mostrate nel capitolo 3 vengono riscritte considerando due relazioni R_1 e R_2 legate da una condizione di join $JC(R_1,R_2)$; tale condizione di join non viene esplicitamente indicata nelle espressioni.

La proprietà (3) del capitolo 3 diventa

$$\sigma_P(R_1 =\bowtie R_2) = \sigma_P(R_1 \bowtie R_2) \qquad (3.3)$$
if P reject nulls su $S(R_2)$

In modo più generale questa proprietà può essere estesa anche al caso di full outerjoin come segue:

$$\sigma_P(R_1 =\bowtie= R_2) = \sigma_P(R_1 =\bowtie R_2) \qquad (1)$$
se P reject null su $S(R_1)$

$$\sigma_P(R_1 =\bowtie= R_2) = \sigma_P(R_1 \bowtie= R_2) \qquad (2)$$
se P reject null su $S(R_2)$

$$\sigma_P(R_1 =\bowtie= R_2) = \sigma_P(R_1 \bowtie R_2) \qquad (3)$$

se P reject null su $S(R_1)$
e P reject null su $S(R_2)$

Si nota che per avere P reject null su $S(R_1)$ e P reject null su $S(R_2)$ dobbiamo necessariamente avere un predicato P che 'interessa' sia $S(R_1)$ che $S(R_2)$ cioè

$$S(P) \not\subseteq S(R_1),\ S(P) \not\subseteq S(R_2)$$

$$\text{e } S(P) \subseteq (S(R_1) \cup S(R_2))$$

Esempio 4.1.1 *Illustriamo meglio il concetto attraverso la seguente query:*

R1 LEFT JOIN R2 ON T1.A=T2.A WHERE condition predicate

dove condition predicate può essere:

$T2.B$ *IS NOT NULL,*
$T2.B > 3$,
$T2.C <= T1.C$,
$T2.B < 2$ *OR* $T2.C > 1$
Le coondizioni elencate sono null-rejected, quindi la query verrà riscritta come:

$R1$ *INNER JOIN* $R2$ *ON* $T1.A = T2.A$ *WHERE condition predicate*

Applicando la 3.3, abbiamo ottenuto le proprietà (1),(2) e (3).

4.2 Definizione degli Insiemi

In questa sezione, vogliamo dare una intuizione delle semplificazioni/ottimizzazioni proposte nella tesi. I metodi di semplificazione/ottimizzazione proposti nella tesi si basano sul concetto di NULL REJECTION proposto da Galindo Legaria e discusso nella sezione precedente. Nella tesi il concetto di NULL REJECTION è stato esteso per considerare le particolarità proprie di un sistema a mediatore; inoltre verrà introdotto un concetto duale, che chiameremo NOT NULL PROJECTION che riguarda gli attributi select di una query e serve a considerare come valide solo quelle tuple in cui almeno un attributo select è non nullo, ovvero ad eliminare dal risultato della query quelle tuple in cui tutti i valori degli attributi di select sono null. Consideriamo la classe globale AUTOMOBILISTI, il suo schema viene descritto mediante la sua mapping table, come segue:

AUTOMOBILISTI	ANAGRAFE	PATENTE	ASSICURAZIONE
Cognome(Join)	Cognome	P_cognome	AS_cognome
Nome (Join)	Nome	P_nome	AS_nome
Età	Età	–	–
StatoCivile	Scivile	–	–
Attività	Professione	–	–
Città	Città	–	–
Via	Via	–	–
Compania	–	–	Compania
N_patente	–	numero	Numpatente
Marca	–	–	TipoAuto
N_targa	–	–	Numtarga
Motorizzazione	–	Mt	–
DataRilascio	–	DataRilascio	–
Validità	–	–	Periodo

Sullo schema globale AUTOMOBILISTI viene fatto la seguente interrogazione:

Cercare l'indirizzo di tutti gli automobilisti di 26 anni che hanno una macchina di tipo Fiat. La query in SQL viene scritta come segue:

```
select Nome, Cognome, Città, Via
from Automobilisti
where TipoAuto='Fiat' and Età='26'
```

Noi imponiamo una condizione sugli attributi di select: le tuple i cui attributi di select sono tutti nulli vanno escluse.

Ciò significa:

```
Nome is not null
and Cognome is not null
and Città is not null
and Via is not null
```

i predicati sono atomici:

P_α = <TipoAuto="Fiat"> e P_β = <Età="26">.

Non sono ammessi predicati del tipo: $P_i =< IsNull(A) >$ con A un attributo.

Allo scopo di individuare le classi locali coinvolte nelle query, si definiscono i seguenti insiemi: T l'insieme degli attributi presenti nella clausola WHERE:

$$T = S(P_1) \cup S(P_2) = \{TipoAuto,Età\}$$

L'insieme degli attributi della proiezione è:

$$\mathcal{A} = \{Nome,Cognome,Città,Via\}$$

l'insieme delle classi coinvolte nella query:

$$\pounds = \pounds_T \cup \pounds_{\mathcal{A}} = \{ANAGRAFE,ASSICURAZIONE,PATENTE\}$$

con

$$\pounds_T = \{ANAGRAFE,ASSICURAZIONE\},\ \pounds_{\mathcal{A}} = \{ANAGRAFE,PATENTE\}$$

$\pounds_T$ è l'insieme delle classi locali che hanno almeno un attributo in T.
$\pounds_{\mathcal{A}}$ è l'insieme delle classi locali che hanno almeno un attributo in $\mathcal{A}$.

Formalmente diamo una definizione degli insiemi e concetti appena presentati sopra.

Definizione 4.2.1 [*£ di G*] *Data una classe globale G*

£ è l'insieme delle sue classi locali, con:

$$\bigcup_{L \in \pounds} S(L) = S(G)$$

Definizione 4.2.2 [*T*]

Data una condizione $P_1 \wedge P_2 \wedge \ldots \wedge P_k$*, dove*
$P_1, P_2, \ldots, P_k$ *sono predicati atomici, sia* $S(P_i)$ *lo schema di* P_i*, con*
$S(P_i) \subseteq S(G)$ *e* $S(P_i) \cap S(P_j) = \emptyset$ *per* $i \neq j$*.*
Definiamo l'insieme T come segue:

$$T = \bigcup_{i=1}^{k} S(P_i)$$

Ovviamente

$$T \subseteq \bigcup_{L \in \pounds} S(L)$$

Definizione 4.2.3 $\pounds_T$

$$\pounds_T = \{Li \in \pounds \mid S(Li) \cap T \neq \emptyset\}$$

Definizione 4.2.4 *T-completo*

Un insieme $X \subseteq \pounds$ *è T-completo rispetto a T o T-completo se*

$$T \subseteq \bigcup_{L \in X} S(L)$$

ovviamente, gli insiemi $\pounds$ *e* $\pounds_T$ *sono T-completi.*

Un insieme $X \subseteq \pounds$ *T-completo è minimale se e solo se non esiste* $Y \subset X$ *con Y T-completo.*

Definizione 4.2.5 $\pounds_{\mathcal{A}}$

$\mathcal{A}$ *è l'insieme degli attributi di S(G) presenti nella clausola SELECT*

$$\pounds_{\mathcal{A}} = \{Li \in \pounds \mid S(Li) \cap \mathcal{A} \neq \emptyset\}$$

Osservazione 4.2.1 *Dalle definizioni di* $\pounds_T$ *e* $\pounds_{\mathcal{A}}$*, ci possono esistere classi appartenenti sia a* $\pounds_T$ *che a* $\pounds_{\mathcal{A}}$*.In alcuni casi,si possono avere* $\pounds_{\mathcal{A}} \subseteq \pounds_T$ *oppure* $\pounds_{\mathcal{A}} \cap \pounds_T \neq \emptyset$ *oppure* $\pounds_{\mathcal{A}} \cap \pounds_T = \emptyset$

Definizione 4.2.6 $\pounds_{P_i} \ldots \pounds_{P_n}$

Dato $\pounds_T$*, sia* $= \mathcal{P}\langle \pounds_{P_1}, \pounds_{P_2}, \pounds_{P_3} \rangle$ *una sequenza definita come*

$$\pounds_{P_i} = \{L \in \pounds_T \mid S(L) \cap S(P_i) = S(P_i) \ e \ L \notin \bigcup_{j=i}^{n} \pounds_{P_{j-1}}\}$$

$\mathcal{P}$ *è una partizione di* $\pounds_T$ *e viene denotato come segue* $\mathcal{P}(\pounds_T)$

Esempio 4.2.1 *Consideriamo che abbiamo la seguente condizione nella clausola WHERE:* $P_\alpha \wedge P_\beta \wedge P_\gamma$ *Abbiamo :* $S(P_\alpha) = A$, $S(P_\beta) = B$, *e* $(P_\gamma) = C$
Abbiamo le seguenti classi:$L1(A,E,K)$, $L2(A,B,K)$, $L3(B,D,K)$, $L4(B,C,K)$, *e* $L5(C,F,K)$

Calcoliamo una partizione di $\pounds_T$*, la nostra seguenza è* $\pounds_{P_\alpha}$, $\pounds_{P_\beta}$, $\pounds_{P_\gamma}$*:*

$\pounds_{P_\alpha} = \{L1,L2\}$
$\pounds_{P_\beta} = \{L \in \pounds_T \mid S(L) \cap S(P_\beta) = S(P_\beta) \ e \ L \notin \pounds_{P_\alpha}\} = \{L3,L4\}$
$\pounds_{P_\gamma} = \{L \in \pounds_T \mid S(L) \cap S(P_\gamma) = S(P_\gamma) \ e \ L \notin (\pounds_{P_\alpha} \cup \pounds_{P_\beta})\} = \{L5\}$

Definizione 4.2.7 $\mathcal{H}_T$

$$\mathcal{H}_T = \{L \in \pounds_T \mid T \subseteq S(L)\}$$

4.3 Proprietà di semplificazione

In questa sezione,l'obiettivo è quello di individuare alcune equivalenze tra espressioni che consentano la semplificazione dell'operatore di full outerjoin. Per semplificazione, intendiamo la possibilità di eliminare alcuni termini coinvolti nell'espressione di outerjoin sia la possibilità di sostituire l'operazione di full outerjoin con un'operazione di left outerjoin, di right outerjoin oppure di join. In entrambi i casi queste operazioni risultano in una ottimizzazione dell'operazione.

Naturalmente le 'classiche norme' di ottimizzazione algebrica verranno seguite, quali ad esempio quella di effettuare il più possibile il 'push' delle condizioni sulle classi locali: tale operazione si può considerare sicuramente come un'ottimizzazione, valida indipendente dal modello di costo adottato, soprattutto nel contesto di un sistema distribuito come quello a mediatore; in tale riguardo un'altra importante operazione è quella di effettuare anche il 'push' della proiezione sulle classi locali (per diminuire la cosidetta arità della relazione risultato, ovvero il numero di attributi che costituiscono la relazione risultato) in quanto anche tale operazione comporta una diminuzione della cardinalità della risposta locale. Per una migliore presentazione dei risultati raggiunti nella tesi, in questo capitolo verranno presentati i risultati senza tenere in considerazione questo aspetto (ad eccezione di alcune note che mettono in risalto la possibilità di tale semplificazione). In un successivo capitolo verranno presentati a parte le semplificazioni possibili sullo schema delle classi locali.

Intuitivamente il nostro approccio è quello di riuscire a togliere quelle tuple che non faranno mai parte del risultato mentre procediamo nell'elaborazione così da evitare di portare avanti dati non rilevanti per la nostra query.

Le proprietà di semplificazione mostrate in seguito valgono sotto le seguenti ipotesi:

Ipotesi fatte:

1. $\pounds$ è full connected: $\pounds = \{L1,L2,...,Ln\}$, $\forall Li,Lj \; \exists \; JC(Li,Lj)$
2. Join consistenti ;
3. Omogeneità semantica

Il punto di partenza è l'espressione FDExpr per il calcolo della Full Disjunction introdotta in sezione 2.3.2.
Riportiamo tale definizione in Algebra relazionale introducendo l'operatore $FJ_{\pounds}$, che quindi corrisponderà ad un caso particolare di Full Disjunction (FD).
Nel seguito parleremo solo di $FJ_{\pounds}$.

In Algebra relazionale $FJ_{\pounds}$ è definita induttivamente come segue:

$$FJ_{\{L1\}\cup\{L2\}} = FJ_{\{L1,L2\}} = L1 =\bowtie= L2$$

dove il full join è fatto su JC(L1,L2).

$$FJ_{\{L1,L2\}\cup\{L3\}} = FJ_{\{L1,L2\}} =\bowtie= L3$$

dove il full join è sulla condizione $JC(L1,L3) \vee JC(L2,L3)$.

$$FJ_{\pounds} = FJ_{\{L1,\ldots,Ln-1\}\cup\{Ln\}} = FJ_{\{L1,\ldots,Ln-1\}} =\bowtie= Ln$$

dove il full join è definito da $JC(L1,Ln) \vee JC(L2,Ln) \vee \cdots \vee JC(Ln-1,Ln)$.

Si può verificare che:

$$FJ_{\{L1,L2\}\cup\{L3\}} = FJ_{\{L1,L3\}\cup\{L2\}}$$

ovvero l'ordine seguito per calcolare $FJ_{\pounds}$ è irrilevante.

$$\begin{aligned} FJ_{\{L1,L2\}\cup\{L3\}} &= (L1 =\bowtie= L2) =\bowtie= L3 \\ &= (L1 =\bowtie= L3) =\bowtie= L2 \\ &= FJ_{\{L1,L3\}\cup\{L2\}} \end{aligned}$$

Le proprietà di semplificazione che verranno discusse dipendono ovviamente dagli attributi presenti nel predicato where e nella select list dell'interrogazione globale. Si possono individuare una serie di condizioni che se verificate consentono particolari semplificazioni; tali condizioni si possono esprimere sulla base degli insiemi definiti in precedenza. La figura 4.1 riassume tali condizioni, individuando 6 casi che corrispondono appunto a particolari semplificazioni. Nel seguito discuteremo questi 6 casi.

I risultati ottenuti verranno mostrati prima in modo intuitivo considerando una classe globale G con 3 classi locali e fornendone una dimostrazione.Quindi, questi risultati verranno generalizzati al caso di una classe globale G con n classi locali.

La classe globale G ha la seguente mapping table:

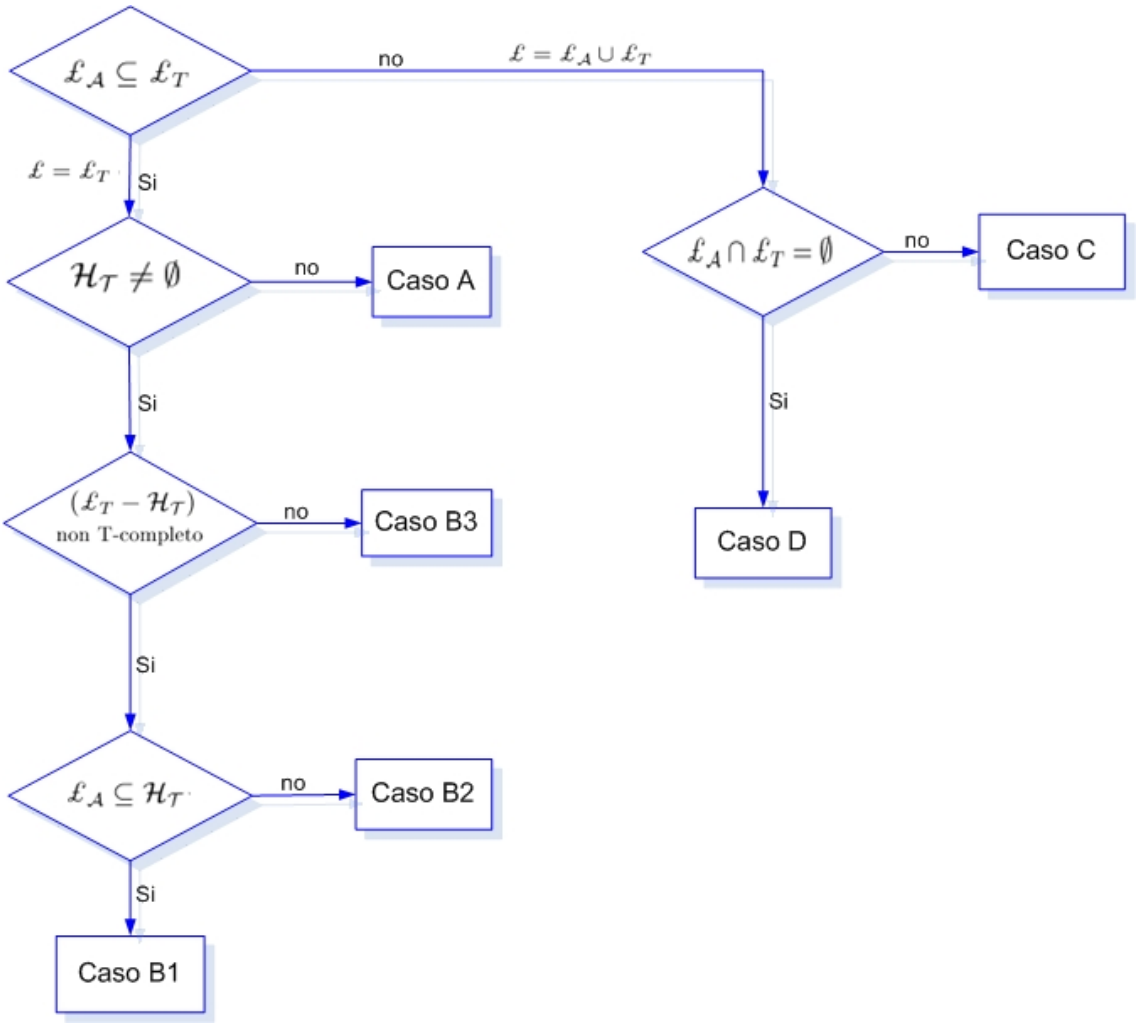

Figura 4.1. Schema del piano di elaborazione di FJ

G	$L1$	$L2$	$L3$
A	A	A	A
B	B	–	–
C	C	–	–
D	D	–	–
E	–	–	E
F	–	F	–
W	W	–	W
V	–	V	–
Y	–	Y	Y
K	K	K	K

Verrà considerata una query sulla classe G nella forma:

select $< list >$
from G
where $< condition \quad predicate >$

Siccome per l'istanza della classe G usiamo l'operatore $FJ_{\pounds}$ introdotto prima, la query viene riscritta in algebra relazionale con

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})]$$

dove P è il predicato che esprime la condizione 'where' e $\pounds$ sono le classi locali coinvolte nella query.

Nel seguito discuteremo la semplificazione dell'espressione $\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})]$ relativa al termine $X = \sigma_P(FJ_{\pounds_T})$ e quindi riscriveremo l'espressione iniziale come $\Pi_{\mathcal{A}}[\sigma_{F_r}(X')]$ dove X' è la riscrittura di X e F_r è il filtro globale residuo. Siccome il calcolo di F_r viene fatto alla fine ed è semplice (non influenza le nostre semplificazioni), allo scopo di migliorare la leggibilità del tutto, tale applicazione di F_r verrà

omessa, scrivendo $\Pi_{\mathcal{A}}[X']$.

Prima di iniziare il nostro studio, proviamo a vedere im modo intuitivo cosa intendiamo fare, cioè diamo un'idea della nostra teoria di ottimizzazione.

Esempio Introduttivo

Allo scopo di illustrare intuitivamente questi concetti, consideriamo la seguente query sulla classe G introdotta in precedenza.

```
select C, D, F
from G
where B op const1 and V op const2
```

Abbiamo $S(P_\alpha) = \{B\}$, $S(P_\beta) = \{V\}$, e $\mathcal{A} = \{C,D,F\}$
Dopo le fasi di parsing della query e l'analisi della mapping table, si determina che le classi L1(B,C,D,K) e L2(F,V,K) sono quelle coinvolte nella query globale, quindi:

$$\pounds = \pounds_T = \{L1,L2\}, \qquad \pounds_{\mathcal{A}} = \{L1,L2\}$$

La risposta alla query viene data da

$$\Pi_{\{C,D,F\}}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds})]$$

Consideriamo le seguenti istanze per le classi locali:

L1

B	C	D	K
b_1	c_1	d_1	k_1
b_2	c_2	d_2	k_2
b_5	c_5	d_5	k_5

L2

F	V	K
f_1	v_1	k_1
f_3	v_3	k_3

Calcoliamo

$\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds}) = \sigma_{F_r}(\sigma_{P_\alpha}(L1) =\bowtie= \sigma_{P_\beta}(L2))$

$$\sigma_{P_\alpha}(L1) =\bowtie= \sigma_{P_\beta}(L2)$$

L1.B	L1.C	L1.D	L1.K	L2.F	L2.V	L2.K
b_1	c_1	d_1	k_1	f_1	v_1	k_1
b_2	c_2	d_2	k_3	$\perp$	$\perp$	$\perp$
b_5	c_5	d_5	k_5	$\perp$	$\perp$	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	f_3	v_3	k_3

$\surd$

Poi al risultato intermedio ottenuto si applica il filtro risiduo globale $F_r = P_\alpha \wedge P_\beta$:

L1.B	L1.C	L1.D	L1.K	L2.F	L2.V	L2.K
b_1	c_1	d_1	k_1	f_1	v_1	k_1

Siccome $P_\alpha \wedge P_\beta$ reject null su $S(L1)$ e $S(L2)$ allora

$$\begin{aligned} \sigma_{P_\alpha \wedge P_\beta}(\sigma_{P_\alpha}(L1) =\bowtie= \wedge\sigma_{P_\beta}(L2)) &= \sigma_{P_\alpha \wedge P_\beta}(\sigma_{P_\alpha}(L1) \bowtie \wedge\sigma_{P_\beta}(L2)) \\ &= (\sigma_{P_\alpha}(L1) \bowtie \wedge\sigma_{P_\beta}(L2)) \end{aligned}$$

Quindi si ha

$$[\sigma_{P_\alpha}(L1) \bowtie \sigma_{P_\beta}(L2)]$$

L1.B	L1.C	L1.D	L1.K	L2.F	L2.V	L2.K
b_1	c_1	d_1	k_1	f_1	v_1	k_1

Questo esempio mette in evidenza le opportunità di semplificazione/ottimizzazione che si hanno nel caso di condizione espressa come congiunzione di predicati atomici che reject null. Essendo questa una situazione molto frequente nelle query, è giustificato a nostro avviso un' analisi approfondità, analisi che verrà presentata nel seguito.

Illustriamo ora il concetto di NOT NULL PROJECTION. Consideriamo la seguente query:

```
select B, C
from G
where E op const1
```

$S(P_\alpha) = \{E\}$ e $\mathcal{A} = \{B,C\}$

$\pounds = \{L3,L1\}$, $\pounds_T = \{L3\}$, $\pounds_\mathcal{A} = \{L1\}$

Consideriamo le seguenti istanze delle classi L1 e L3:

L1

B	C	K
b_1	c_1	k_1
b_2	c_2	k_2
b_5	c_5	k_5

L3

E	K
e_1	k_1
e_2	k_2
e_3	k_3

$$\sigma_{P_\alpha}(FJ_\pounds) = \sigma_{F_r}(\sigma_{true}(L1) =\bowtie= \sigma_{P_\alpha}(L3)) \quad (4.1)$$
$$\sigma_{true}(L1) =\bowtie= \sigma_{P_\alpha}(L3))$$

L1.B	L1.C	L1.K	L3.E	L3.K	
b_1	c_1	k_1	e_1	k_1	$\surd$
b_2	c_2	k_3	e_3	k_3	$\surd$
b_5	c_5	k_5	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	e_2	k_2	$\surd$

Poi applichiamo al risultato ottenuto il filtro globale residuo $F_r = P_\alpha$:

L1.B	L1.C	L1.K	L3.E	L3.K	
b_1	c_1	k_1	e_1	k_1	$\surd$
b_2	c_2	k_3	e_3	k_3	$\surd$
$\perp$	$\perp$	$\perp$	e_2	k_2	$\surd$

Poi facciamo la proiezione sugli attributi $\Pi_\mathcal{A} = \{B,C\}$, il risultato finale è:

B	C
b_1	c_1
b_3	c_3
$\perp$	$\perp$

Tabella 4.1. risultato senza condizione su gli attributi di selezione

Si ossserva che nella risposta finale alla query, abbiamo tuple che hanno valori nulli su tutti gli attributi della proiezione. Per eliminare dalle risposte tale tuple si impone la seguente condizione:

NOT(B null and C null)

che equivale in base alle leggi di De Morgan a

(B is not null OR C is not null)

Questa condizione viene scritta come predicato $P_{\mathcal{A}}$ =NOT(B null and C null) Per cui la 4.1 viene riscritta come segue:

$$\sigma_{P_\alpha \wedge P_{\mathcal{A}}}(FJ_{\pounds}) = \sigma_{F_r}(\sigma_{P_{\mathcal{A}}}(L1) =\bowtie= \sigma_{P_\alpha}(L3))$$

Essendo $\sigma_{P_\alpha \wedge P_{\mathcal{A}}}$ reject null sia su $S(L1)$ che su $S(L2)$, si ha

$$\begin{aligned}\sigma_{P_\alpha \wedge P_{\mathcal{A}}}(FJ_{\pounds}) &= \sigma_{F_r}(\sigma_{P_{\mathcal{A}}}(L1) =\bowtie= \sigma_{P_\alpha}(L3)) \\ &= \sigma_{F_r}(\sigma_{P_{\mathcal{A}}}(L1) \bowtie \sigma_{P_\alpha}(L3)) \\ &= \sigma_{P_{\mathcal{A}}}(L1) \bowtie \sigma_{P_\alpha}(L3)\end{aligned}$$

L'aggiunta di $P_{\mathcal{A}}$ permette di ottenere:

$$\sigma_{P_\alpha}(L3) \bowtie \sigma_{P_{\mathcal{A}}}(L1)$$

L1.B	L1.C	L1.K	L3.E	L3.K	
b_1	c_1	k_1	e_1	k_1	$\surd$
b_2	c_2	k_3	e_3	k_3	$\surd$

Si nota subito come la terza riga della tabella 4.1 è stata eliminata. Effettuando la proiezione su B e C si ottiene:

B	C
b_1	c_1
b_3	c_3

L'applicazione di tale predicato ($P_{\mathcal{A}}$) ha come effetto di eliminare tutte le tuple che hanno valori nulli sugli attributi di proiezione. Nel nostro studio applicheremo questo vincolo nei casi in cui risulta necessario: in genere nel caso $\pounds_{\mathcal{A}} \cap \pounds_T = \emptyset$.

Caso 1 $\pounds_{\mathcal{A}} \subseteq \pounds_T$

Esaminiamo i vari casi indicati in figura 4.1. Iniziamo con i casi relativi a $\pounds_{\mathcal{A}} \subseteq \pounds_T$. Tale inclusione implica che $\pounds = \pounds_T$ avendo $\pounds = \pounds_T \cup \pounds_{\mathcal{A}}$, ovvero l'insieme di classi coinvolte nella query, coincide con $\pounds_T$, cioè è sufficiente considerare solo le classi coinvolte nella condizione where.

Caso A) $\pounds_{\mathcal{A}} \subseteq \pounds_T$ e $\mathcal{H}_T = \emptyset$

Consideriamo una condizione $P = P_\alpha \wedge P_\beta$ e il seguente insieme $\pounds_T = \{L1,L2,L3\}$. Supponiamo la partizione $\mathcal{P}$ sia

$$\pounds_{P_\alpha} = \{L1, L3\}, \quad \pounds_{P_\beta} = \{L2\}$$

Ometteremo nelle formule di scrivere le classi con i predicati che supportano come abbiamo fatto nei esempi precedenti. Di certo sappiamo in modo generale che un classe L appartenente a sottoinsieme $\pounds_{P_i}$ supporta il predicato P_i e potrebbe supportare altri predicati; una classe L appartenente a $(\pounds_T - \pounds_T \cap \pounds_{P_i})$ non supporta il predicato P_i.

Verrà discusso e dimostrato nel seguito che valgono le seguenti equivalenze

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}\left[\sigma_P((\underbrace{L1 =\bowtie= L3}_{\pounds_{P_\alpha}}) =\bowtie= \underbrace{L2}_{\pounds_{P_\beta}})\right] \tag{4.2}$$

$$= \Pi_{\mathcal{A}}\left[\underbrace{\sigma_{P_\alpha}(L1 =\bowtie= L3)}_{\pounds_{P_\alpha}} \bowtie \underbrace{\sigma_{P_\beta}(L2)}_{\pounds_{P_\beta}}\right] \tag{4.3}$$

La proprietà che otteniamo è quella seguente:

$$\boxed{\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}\left[\sigma_{P_\alpha}(FJ_{\pounds_{P_\alpha}}) \bowtie \sigma_{P_\beta}(FJ_{\pounds_{P_\beta}})\right]} \qquad 4.3a$$

Nei sottoinsiemi $\pounds_{P_\alpha}$ e $\pounds_{P_\beta}$, non si può trovare una classe che soddisfa tutti i predicati (supporta T), in quanto $\mathcal{H}_T = \emptyset$. Con riferimento alla 4.3*a* si può notare come questa espressione è nella forma:
$L' \bowtie L''$, essendo il join commutativo possiamo scrivere:

$$L' \bowtie L'' = L'' \bowtie L'$$

Di conseguenza la 4.3*a* può essere riscritta come segue:

$$\Pi_{\mathcal{A}} \left[\sigma_{P_\alpha}(FJ_{\pounds_{P_\alpha}}) \bowtie \sigma_{P_\beta}(FJ_{\pounds_{P_\beta}})\right] = \Pi_{\mathcal{A}} \left[\sigma_{P_\beta}(FJ_{\pounds_{P_\beta}}) \bowtie \sigma_{P_\alpha}(FJ_{\pounds_{P_\alpha}})\right] \quad (4.4)$$

Essendo $card(\pounds_{P_\alpha}) > 1$, allora

$$\Pi_{\mathcal{A}} \left[\sigma_{P_\beta}(FJ_{\pounds_{P_\beta}}) \bowtie \sigma_{P_\alpha}(FJ_{\pounds_{P_\alpha}})\right] = \Pi_{\mathcal{A}} \left[\sigma_{P_\beta}(FJ_{\pounds_{P_\beta}}) \bowtie (\sigma_{P_\alpha}(L1) =\bowtie= \sigma_{P_\alpha}(L3))\right]$$

Nel seguito verrà dimostrato che è possibile riscrivere la precedente espressione come

$$\Pi_{\mathcal{A}} \left[\sigma_{P_\beta}(FJ_{\pounds_{P_\beta}}) \bowtie \sigma_{P_\alpha}(FJ_{\pounds_{P_\alpha}})\right] = \Pi_{\mathcal{A}} \left[\left(\sigma_{P_\beta}(FJ_{\pounds_{P_\beta}}) =\bowtie \sigma_{P_\alpha}(L1)\right) =\bowtie \sigma_{P_\alpha}(L3)\right] \quad (4.5)$$

Dimostriamo la proprietà 4.5.

Consideriamo

$$\sigma_{P_\alpha \wedge P_\beta}(FJ_{\{L1,L2,L3\}})$$

Abbiamo secondo quanto detto all'inizio delle nostre ipotesi

$$FJ_{\{L1,L2,L3\}} = FJ_{\{L2,L3\}} =\bowtie= FJ_{\{L1\}} = FJ_{\{L2,L1\}} =\bowtie= FJ_{\{L3\}}$$

Poi, abbiamo:

$$\begin{aligned} FJ_{\{L1,L2,L3\}} &= FJ_{\{L2,L1\}} =\bowtie= FJ_{\{L3\}} \\ &= ((L2 =\bowtie= L1) =\bowtie= L3) \end{aligned} \quad (4.6)$$

Dalla 4.6, determiniamo ora:

$$\sigma_{P_\alpha \wedge P_\beta}(FJ_{\{L1,L2,L3\}}) = \sigma_{P_\alpha \wedge P_\beta}((L2 =\bowtie= L1) =\bowtie= L3) \quad (4.7)$$

Consideriamo la proprietà 3 di Legando-legaria della sezione 4.1, riscritta con i predicati: P_α e P_β

$\sigma_{P_\alpha \wedge P_\beta}(R_1 =\bowtie= R_2) = \sigma_{P_\alpha \wedge P_\beta}(R_1 \bowtie R_2)$

se $(S(P_\alpha) \cup S(P_\beta)) \cap S(R_1) = S(P_\alpha)$ e $(S(P_\alpha) \cup S(P_\beta)) \cap S(R_2) = S(P_\beta)$.

Con riferimento alla 4.7, calcoliamo

$$\sigma_{P_\alpha \wedge P_\beta}((L2 =\bowtie= L1) =\bowtie= L3) \tag{4.8}$$

per effettuare l'operazione consideriamo dapprima il calcolo di $\sigma_{P_\alpha \wedge P_\beta}(L2 =\bowtie= L1)$:

Essendo $\mathcal{H}_T = \emptyset$, cio implica che nessuna delle classi L2 e L1 supportano il predicato $P = P_\alpha \wedge P_\beta$, quindi applicando la proprietà 3 di Legando appena citata, abbiamo:

$$\sigma_{P_\alpha \wedge P_\beta}(L2 =\bowtie= L1) = \sigma_{P_\alpha \wedge P_\beta}(L2 \bowtie L1)$$

quindi La 4.8 viene riscritta come segue:

$$\sigma_{P_\alpha \wedge P_\beta}((L2 =\bowtie= L1) =\bowtie= L3) \quad = \quad \sigma_{P_\alpha \wedge P_\beta}((L2 \bowtie L1) =\bowtie= L3) \tag{4.9}$$

Poniamo $L^{'} = L2 \bowtie L1$, pertanto possiamo scrivere $\sigma_{P_\alpha \wedge P_\beta}(L^{'} =\bowtie= L3)$

Consideriamo ora la proprietà 1 di Legando Legaria della sezione 4.1, riscritta come:

$$\sigma_{P_\alpha \wedge P_\beta}(R_1 =\bowtie= R_2) = \sigma_{P_\alpha \wedge P_\beta}(R_1 =\bowtie R_2)$$

Se $P_\alpha \wedge P_\beta$ reject null in R_2, cioè $S(P_\beta) \cap S(R_2) = \emptyset$ implica che tutte le tuple di R_2 che non fanno match con R_1 verranno eliminate visto che gli attributi delle tuple di R_2 hanno valori null su $S(P_\beta)$.

Di conseguenza:

$$\sigma_{P_\alpha \wedge P_\beta}(L^{'} =\bowtie= L3) = \sigma_{P_\alpha \wedge P_\beta}(L^{'} =\bowtie L3)$$

Riassumendo possiamo scrivere la 4.8 come segue:

$$\sigma_{P_\alpha \wedge P_\beta}((L2 =\bowtie= L1) =\bowtie= L3) \quad = \quad \sigma_{P_\alpha \wedge P_\beta}((L2 \bowtie L1) =\bowtie L3) \tag{4.10}$$

Pertanto riscriviamo la nostra espressione iniziale:

$$\sigma_{P_\alpha \wedge P_\beta}(FJ_{\{L1,L2,L3\}}) \quad = \quad \sigma_{P_\alpha \wedge P_\beta}(FJ_{\{L1\}} =\bowtie= FJ_{\{L2,L3\}}) \tag{4.11}$$

$$= \quad \sigma_{P_\alpha \wedge P_\beta}((L2 \bowtie L1) =\bowtie L3) \tag{4.12}$$

Però possiamo anche scrivere:

$$FJ_{\{L1,L2,L3\}} = FJ_{\{L2,L3\}} =\bowtie= FJ_{\{L1\}}$$

ed in modo analogo ottenere che

$$\sigma_{P_\alpha \wedge P_\beta}((L2 =\bowtie= L3) =\bowtie= L1) = \sigma_{P_\alpha \wedge P_\beta}((L2 \bowtie L3) =\bowtie L1)$$

e quindi:

$$\begin{aligned} \sigma_{P_\alpha \wedge P_\beta}(FJ_{\{L1,L2,L3\}}) &= \sigma_{P_\alpha \wedge P_\beta}(FJ_{\{L2,L3\}} =\bowtie= FJ_{\{L1\}}) \\ &= \sigma_{P_\alpha \wedge P_\beta}((L2 \bowtie L3) =\bowtie L1) \end{aligned} \quad (4.13)$$

Considerando la 4.12 e la 4.13 possiamo scrivere:

$$\sigma_{P_\alpha \wedge P_\beta}((L2 \bowtie L3) =\bowtie L1) = \sigma_{P_\alpha \wedge P_\beta}((L2 \bowtie L1) =\bowtie L3)$$

Questa uguaglianza è assurda, l'unica soluzione possibile e accettabile è:

$$\sigma_{P_\alpha \wedge P_\beta}((L2 =\bowtie L3) =\bowtie L1) = \sigma_{P_\alpha \wedge P_\beta}((L2 =\bowtie L1) =\bowtie L3)$$

Essendo $\pounds_{P_\beta} = \{L2\}$, L'ugualianza si può anche riscrivere come segue:

$$((\sigma_{P_\beta}(FJ_{\pounds_{P_\beta}}) =\bowtie \sigma_{P_\alpha}(L1)) =\bowtie \sigma_{P_\alpha}(L3)) = ((\sigma_{P_\beta}(FJ_{\pounds_{P_\beta}}) =\bowtie \sigma_{P_\alpha}(L3)) =\bowtie \sigma_{P_\alpha}(L1))$$

Di conseguenza la $\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})]$ può essere riscritta come segue :

$$\boxed{\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}\left[\left(\sigma_{P_\beta}(FJ_{\pounds_{P_\beta}}) =\bowtie \sigma_{P_\alpha}(L1)\right) =\bowtie \sigma_{P_\alpha}(L3)\right]}$$

Dimostrazione della proprietà 4.3a

Consideriamo la seguente query:

$$\Pi_{\mathcal{A}}\left[\sigma_{P_\alpha \wedge P_\beta}(L1 =\bowtie= L2 =\bowtie= L3)\right]$$

$P = P_\alpha \wedge P_\beta$ e $\pounds_T = \{L1,L2,L3\}$ e $\pounds_{\mathcal{A}} = \{L1,L2,L3\}$ La partizione dell'insieme $\pounds_T$ nella seguenza $\pounds_{P_\alpha}$ e $\pounds_{P_\beta}$ è

$\pounds_{P_\alpha} = \{L1,L3\}$ e $\pounds_{P_\beta} = \{L2\}$

$$\mathcal{H}_T = \emptyset \Rightarrow \sigma_P(L2) = \sigma_P(L1) = \sigma_P(L3) = \emptyset, \quad \text{con} \quad P = P_\alpha \wedge P_\beta \qquad (i)$$

Ma $L1$,$L3 \in \pounds_{P_\alpha} \Rightarrow \sigma_{P_\alpha}(L1) \neq \emptyset$ e $\sigma_{P_\alpha}(L3) \neq \emptyset$ (ii)

Quindi (i) e (ii) implicano che $\sigma_{P_\alpha \wedge P_\beta}(L1 =\bowtie= L3) = \emptyset$ (iii)

$L2 \in \pounds_{P_\beta} \Rightarrow \sigma_{P_\beta}(L2) \neq \emptyset$ (iv)

Allora (iv) e (iii) implicano che riscriviamo la query come segue:

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(L1 =\bowtie= L2 =\bowtie= L3)] = \Pi_{\mathcal{A}}[\sigma_{P_\alpha}(L1 =\bowtie= L3) \bowtie \sigma_{P_\beta}(L2)]$$

Visto che $card(\pounds_{P_\alpha}) > 1$, si ha

$$\Pi_{\mathcal{A}}[\sigma_{P_\beta}(L2) \bowtie \sigma_{P_\alpha}(L1 =\bowtie= L3)] = \Pi_{\mathcal{A}}[(\sigma_{P_\beta}(L2) =\bowtie \sigma_{P_\alpha}(L1)) =\bowtie \sigma_{P_\alpha}(L3)]$$

Esempio 4.3.1

Come esempio della situazione ipotizzata nella dimostrazione, consideriamo la seguente query:

```
select A, E
from G
where W op const1 and V op const2
```

Quindi $\pounds_T = \{L1,L2,L3\}$ e $\pounds_{\mathcal{A}} = \{L1,L2,L3\}$

$S(P_\alpha) = \{W\}$, e $S(P_\beta) = \{V\}$,per cui abbiamo:

$\pounds_{P_\alpha} = \{L1,L3\}$ e $\pounds_{P_\beta} = \{L2\}$

Consideriamo le seguenti tabelle:

$L1$

A	W	K
a_1	w_1	k_1
a_2	w_2	k_2
a_3	w_3	k_3

$L2$

A	V	K
a_1	v_1	k_1
a_4	v_4	k_4
a_5	v_5	k_5

$L3$

A	E	W	K
a_1	e_1	w_1	k_1
a_3	e_3	w_3	k_3
$\perp$	e_6	w_6	k_6

Il calcolo di $\sigma_P(FJ_{\pounds_T})$ nella sua versione naive ci dà la tabella seguente:

L1.A	L1.W	L1.K	L2.A	L2.V	L2.K	L3.A	L3.E	L3.W	L3.K	
a_1	w_1	k_1	a_1	v_1	k_1	a_1	e_1	w_1	k_1	$\surd$
a_2	w_2	k_2	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	
a_3	w_3	k_3	$\perp$	$\perp$	$\perp$	a_3	e_3	w_3	k_3	
$\perp$	$\perp$	$\perp$	a_4	v_4	k_4	$\perp$	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	a_6	v_6	k_6	$\perp$	e_6	w_6	k_6	$\surd$

Applichiamo la seguente semplificazione

$\sigma_P(FJ_{\pounds_T}) = (\sigma_{P_\beta}(L2) \bowtie \sigma_{P_\alpha}(L1 =\bowtie= L3))$,

ma bisogna calcolare prima $(\sigma_{P_\alpha}(L1) =\bowtie= \sigma_{P_\alpha}(L3))$

L1.A	L1.W	L1.K	L3.A	L3.E	L3.W	L3.K
a_1	w_1	k_1	a_1	e_1	w_1	k_1
a_2	w_2	k_2	$\perp$	$\perp$	$\perp$	$\perp$
a_3	w_3	k_3	a_3	e_3	w_3	k_3
$\perp$	$\perp$	$\perp$	$\perp$	e_6	w_6	k_6

A questo punto possiamo calcolare $\sigma_P(FJ_{\pounds_T}) = \sigma_P(L2 \bowtie (L1 =\bowtie= L3))$

$$[\sigma_{P_\beta}(L2) \bowtie \sigma_{P_\alpha}(L1 =\bowtie= L3)]$$

L2.A	L2.V	L2.K	L1.A	L1.W	L1.K	L3.A	L3.E	L3.W	L3.K
a_1	v_1	k_1	a_1	w_1	k_1	a_1	e_1	w_1	k_1
a_6	w_6	k_6	$\perp$	$\perp$	$\perp$	$\perp$	e_6	w_6	k_6

L'espressione $\sigma_P FJ_{\pounds_T} = [\sigma_{P_\beta}(L2) \bowtie \sigma_{P_\alpha}(L1 =\bowtie= L3)]$ può essere anche riscritta come segue

$$[(\sigma_{P_\beta}(L2) =\bowtie \sigma_{P_\alpha}(L1)) =\bowtie \sigma_{P_\alpha}(L3)]$$

L1.A	L1.W	L1.K	L2.A	L2.V	L2.K	L3.A	L3.E	L3.W	L3.K	
a_1	w_1	k_1	a_1	v_1	k_1	a_1	e_1	w_1	k_1	$\surd$
$\perp$	$\perp$	$\perp$	a_4	v_4	k_4	$\perp$	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	a_6	v_6	k_6	$\perp$	e_6	w_6	k_6	$\surd$

e questo rappresenta un'altra forma di ottimizzazione. I risultati ottenuti come le due semplificazioni sono diversi in quanto nella seconda ottimizzazione abbiamo ancora tuple irrilevanti. Però bisogna notare che nei calcoli intermedi la prima semplificazione può dare dei risultati intermedi grandi in termini di numero di tuple: basta guardare il calcolo di $(\sigma_{P_\alpha}(L1) =\bowtie= \sigma_{P_\alpha}(L3))$. Mentre la seconda ottimizzazione può mantenere il numero delle tuple costanti basta guardare il calcolo delLa $(\sigma_{P_\beta}(L2) =\bowtie \sigma_{P_\alpha}(L1)) =\bowtie \sigma_{P_\alpha}(L3))$. Evidentemente nel caso in cui L2 risulta abbastanza grande allora si potrebbe ottenere un peggioramento della nostra ottimizzazione perché avremo il numero intermedio di tuple sempre grande in termine di cardinalità.

Proprietà generale nel caso di n classi

Consideriamo i seguenti insiemi:

$\pounds_T = \{L1,...,Ln\}$, consideriamo una sua partizione rispetto ai predicati P_α e P_β:

$\pounds_{P_\alpha} = \{L1,...,Lk\}$ e $\pounds_{P_\beta} = \{L(k+1),...,Ln\}$

Allora possiamo scrivere:

$$\Pi_{\mathcal{A}}\left[\sigma_P(FJ_T)\right] = \Pi_{\mathcal{A}}\left[\sigma_P(\underbrace{L1 =\bowtie= \cdots =\bowtie= Lk}_{\pounds_{P_\alpha}} =\bowtie= \underbrace{L(k+1) =\bowtie= \cdots =\bowtie= Ln}_{\pounds_{P_\beta}})\right]$$

$$= \Pi_{\mathcal{A}}[\sigma_{P_\alpha}(FJ_{\pounds_{P_\alpha}}) \bowtie \sigma_{P_\beta}(FJ_{\pounds_\beta})] \quad (4.14)$$

$$= \Pi_{\mathcal{A}}\left[\left(\left(\sigma_{P_\alpha}(FJ_{\pounds_{P_\alpha}}) =\bowtie \sigma_{P_\beta}(L(k+1))\right) =\bowtie= \cdots\right) =\bowtie= Ln\right] \quad (4.15)$$

Osservazione 4.3.1

Riprendiamo la (4.14), se $\pounds_T$ fosse minimale (si guarda la definizione (4.2.3)), la (4.14) risulterebbe la proprietà adatta nel caso di minimalità perché ci sarebbe al massimo una unica classe per ogni predicato e otterrebbe una espressione semplificata composta da soli operatori di join fra le classi.Inoltre il calcolo del filtro globale F_r è semplice. In particolare, nel caso in cui $\pounds_T$ è minimale, risulta essere $F_r = true$, cioè non occorre applicare nessuna condizione dopo il calcolo di FJ. Questa considerazione implica che gli attributi relativi alla condizione di where vanno eliminati dagli schemi delle classi locali coinvolte nella query considerata.

In generale, $card(\pounds_T) \leq card(T)$ è la condizione necessaria per avere $\pounds_T$ minimale, e la definizione della minimalità (4.2.3) è la condizione sufficiente.In condizione di minimalità, l'espressione semplificata di $FJ_{\pounds_T}$ è composta da soli operatori di join.

Dimostrazione:

Data una query con N predicati. Il numero **massimo** di partizione ottenibile è N (un insieme per ogni predicato):

$$\pounds_{P_1}, \ldots, \pounds_{P_n}$$

Avendo $\mathcal{H}_T = \emptyset$, possiamo applicare la 4.14:

$$\sigma_{P_1 \wedge \cdots \wedge P_n}(FJ_T) = \sigma_{P_1}(FJ_{\pounds_{P_1}}) \bowtie \cdots \bowtie \sigma_{P_n}(FJ_{\pounds_{P_n}}) \qquad (I)$$

Supponiamo:

$$card(\pounds_{P_1}) = \cdots = card(\pounds_{P_n}) = 1$$

allora ogni $\pounds_{P_i}$ contiene una unica classe L_i, Per cui la (I) diventa:

$$\sigma_{P_1 \wedge \cdots \wedge P_n}(FJ_T) = \sigma_{P_1}(L1) \bowtie \cdots \bowtie \sigma_{P_n}(Ln) \qquad (II)$$

Si osserva che l'espressione semplificata di $FJ_{\pounds_T}$ è composta da soli operatori di join per cui $\pounds_T$ è minimale.
Si calcola:

$$card(\pounds_T) = \Sigma_{i=1}^{N} card(\pounds_{P_i}) = card(\pounds_{P_1}) + \cdots + card(\pounds_{P_n}) = N$$

Per definizione

$$card(T) = N$$

Sapendo che abbiamo al massimo N partizione per N predicati (si può avere un insieme vuoto), allora scriviamo:

$$card(\pounds_T) \leq card(T)$$

condizione necessaria per avere $\pounds_T$ minimale.

Proprietà generale in caso di minimalità di $\pounds_T$:

Nel processo di semplificazione, se $card(\pounds_T) \leq card(T)$ e $\pounds_T - \{L\}$ è non T-completo con $L \in \pounds_T$, allora l'espressione di $FJ_{\pounds_T}$ è composta da soli operatori di join. Gli attributi relativi alla condizione di where vanno eliminati dagli schemi delle classi locali coinvolte nella query considerata.

Esempio 4.3.2

Consideriamo una query con tre predicati: $S(P_\alpha) = A$, $S(P_\beta) = B$ e $S(P_\gamma) = C$

In caso di minimalità si può avere al massimo tre classi. Illustriamo alcune configurazione:
- tre classi: l'unica configurazione possibile è: L1(A,..,K),L2(B,..,K) e L3(C,..,K)
- due classi: L1(A,C) e L2(C,B),oppure L1(A,B) e L2(B,C), L1(A,B) e L2(A,C), L1(A,B) e L2(C) ..ecc
-una classe: L1(A,B,C).

- **Caso B1)** $\mathcal{H}_\mathcal{T} \neq \emptyset$, $(\pounds_T - \mathcal{H}_\mathcal{T})$ non T-completo, e $\pounds_\mathcal{A} \subseteq \mathcal{H}_\mathcal{T}$

  In questo caso le uniche classi da elaborale sono quelle di $\mathcal{H}_\mathcal{T}$ tramite full outerjoin.
  Consideriamo $P = P_\alpha \wedge P_\beta$ e che le classi L2 e L3 supportano solo il predicato P_α mentre L1 supporta $P_\alpha \wedge P_\beta$.
  Quindi

$$\mathcal{H}_\mathcal{T} = \{L1\}, \;\; \pounds_T = \{L1,L2,L3\}$$

Nel seguito verrà dimostrato che

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}\left[\sigma_P((\underbrace{L1}_{\mathcal{H}_\mathcal{T}}) =\bowtie= \underbrace{(L2 =\bowtie= L3)}_{\pounds_T - \mathcal{H}_\mathcal{T}})\right] \quad (4.16)$$

$$= \Pi_{\mathcal{A}}\left[\sigma_{P_\alpha \wedge P_\beta}(\underbrace{L1}_{\mathcal{H}_\mathcal{T}}) =\bowtie= \underbrace{\sigma_{P_\alpha}(L2 =\bowtie= L3)}_{\pounds_T - \mathcal{H}_\mathcal{T}}\right] \quad (4.17)$$

$$= \Pi_{\mathcal{A}}\left[\sigma_P(\underbrace{L1}_{\mathcal{H}_\mathcal{T}})\right] \quad (4.18)$$

$$= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\mathcal{H}_T})\right] \quad (4.19)$$

La proprietà ottenuta è la seguente:

$$\boxed{\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\mathcal{H}_T})\right]}$$

Si nota che in questo caso le classi L2 e L3 vengono eliminate nella calcolo di $\sigma_P(FJ_{\pounds_T})$.

Dimostrazione

Consideriamo

$$\Pi_{\mathcal{A}}[\sigma_P(L2 =\bowtie= L3 =\bowtie= L1)]$$

Con $\pounds_{\mathcal{A}} = \{L1\}$, $\mathcal{H}_T = \{L1\}$ quindi $\pounds_{\mathcal{A}} \subseteq \mathcal{H}_T$.

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}[\sigma_P(FJ_{\mathcal{H}_T})]$$

Inoltre supponiamo che $(\pounds_T - \mathcal{H}_T)$ è non T-completo

ovvero possiamo individuare P_α e P_β tale che $P = P_\alpha \wedge P_\beta$

$$\sigma_{P_\alpha}(L2 =\bowtie= L3) \neq \emptyset$$

e

$$\sigma_{P_\alpha \wedge P_\beta}(L2 =\bowtie= L3) = \emptyset \quad (i)$$

$$L1 \in \mathcal{H}_T \qquad \Rightarrow \qquad \sigma_{P_\alpha \wedge P_\beta}(L1) \neq \emptyset \tag{ii}$$

(i) e (ii) implicano la seguente semplificazione:

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(L2 =\bowtie= L3 =\bowtie= L1)] = \Pi_{\mathcal{A}}[\sigma_{P_\alpha}(L2 =\bowtie= L3) \bowtie= \sigma_{P_\alpha \wedge P_\beta}(L1)] \tag{iii}$$

$\{L2,L3\} \not\subset \pounds_{\mathcal{A}} \Rightarrow \Pi_{\mathcal{A}}(L2) = \emptyset$ e $\Pi_{\mathcal{A}}(L3) = \emptyset$ allora:

$$\Pi_{\mathcal{A}}(\sigma_{P_\alpha}(L2 =\bowtie= L3)) = \emptyset \tag{iv}$$

(iv) implica che:

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha}(L2 =\bowtie= L3) \bowtie= \sigma_{P_\alpha \wedge P_\beta}(L1)] = \Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(L1)] \tag{v}$$

Abbiamo quindi verificato che

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}[\sigma_P(FJ_{\mathcal{H}_T})]$$

Esempio 4.3.3

Consideriamo la seguente interrogazione:

```
select C, D
from G
where A op exp1 and B op exp2
```

$\pounds_T = \{L1,L2,L3\}$, $\mathcal{H}_T = \{L1\}$, $\pounds_{\mathcal{A}} = \{L1\}$

$$\begin{aligned} FJ_{\pounds} &= \Pi_{\{C,D\}}[\sigma_{P_\alpha \wedge P_\beta}(L2 =\bowtie= L3 =\bowtie= L1)] \\ &= \Pi_{\{C,D\}}[\sigma_{P_\alpha}(L2 =\bowtie= L3) \bowtie= \sigma_{P_\alpha \wedge P_\beta}(L1)] \\ &= \Pi_{\{C,D\}}[\sigma_{P_\alpha \wedge P_\beta}(L1)] \\ &= FJ_{\mathcal{H}_T} \end{aligned}$$

$L1$

A	B	C	D	K
a_1	b_1	c_1	d_1	k_1
a_5	b_5	c_5	d_5	k_5

$L2$

A	K
a_1	k_1
a_3	k_3

$L3$

A	K
a_3	k_3
a_5	k_5
a_7	k_7

Prima di tutto calcoliamo la FJ nella sua versione naive:

$$\sigma_{P_\alpha \wedge P_\beta}(L2 =\bowtie= L3 =\bowtie= L1)$$

L1.A	L1.B	L1.C	L1.D	L1.K	L2.A	L2.K	L3.A	L3.K	
a_1	b_1	c_1	d_1	k_1	a_1	k_2	$\perp$	$\perp$	$\surd$
a_5	b_5	c_5	d_5	k_5	$\perp$	$\perp$	a_5	k_5	$\surd$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	a_3	k_3	a_3	k_3	
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	a_7	k_7	

Le tuple con $\surd$ sono quelle che fanno parte della risposta.

$$\sigma_{P_\alpha}(L2 =\bowtie= L3)$$

L2.A	L2.K	L3.A	L3.K
a_1	k_1	$\perp$	$\perp$
a_3	k_3	a_3	k_3
$\perp$	$\perp$	a_5	k_5
$\perp$	$\perp$	a_7	k_7

$\sigma_{P_\alpha \wedge P_\beta}(L2 =\bowtie= L3) = \emptyset$ e $\Pi_{\{C,D\}}[\sigma_{P_\alpha}(L2 =\bowtie= L3)] = \emptyset$

Quindi si può notare che la risposta alla query sia

$$\Pi_{\{C,D\}}[\sigma_{P_\alpha \wedge P_\beta}(L1)]$$

L1.C	L1.D
c_1	d_1
c_5	d_5

Osservazione 4.3.2 *Prima di passare ad enunciare la proprietà nel caso generale, vogliamo aprire una parentesi e far osservare come nel caso in esame, sia evidente un'ulteriore ottimizzazione dovuta al 'push' della proiezione. Alla classe L1 viene applicata l'intera condizione, quindi tutte le tuple risultanti da tale classe soddisfano la condizione; ne consegue che nel risultato della interrogazione su tale classe non è necessario considerare gli attributi relativi alla condizione ma solo gli attributi relativi alla proiezione(in effetti servirebbero anche gli attributi di Join, ma non in questo caso perché c'è una sola classe nell'espressione) cioè C e D: in definitiva si può fare il 'push' della proiezione di C e D su L1 e semplificare lo schema di tale classe.*

Proprietà generale nel caso di n classi

Si parte dalla proprietà

$$[\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] \quad = \quad \Pi_{\mathcal{A}}[\sigma_P(FJ_{\mathcal{H}_T})] \tag{4.20}$$

verificata in precedenza con $\mathcal{H}_T$ contenente un'unica classe.

Nel caso generale si ha $card(\mathcal{H}_T) > 1$ e la proprietà (1) è ancora valida. Inoltre, in questa situazione un'altra importante semplificazione riguarda il termine

$$\sigma_P(FJ_{\mathcal{H}_T})$$

che e $\mathcal{H}_T = \{L1,...,Lk\}$ con $k > 1$

equivale a

$$\sigma_P(L1 =\bowtie= L2 =\bowtie= \cdots =\bowtie= Lk)$$

Siccome tutte le $Li \in \mathcal{H}_T$ supportano la condizione P, tale espressione equivale a

$$\sigma_P(L1) =\bowtie= \sigma_P(L2) =\bowtie= \cdots =\bowtie= \sigma_P(Lk)$$

ovvero può essere fatto il 'push' del predicato P sulle classi locali.
Notare la differenza rispetto al 'push' classico nel caso di join.

$$\sigma_P(L1 \bowtie L2)$$

basta che sia P supportato solo da L1 per scrivere

$$\sigma_P(L1) \bowtie L2$$

invece con l'outerjoin

$$\sigma_P(L1 =\bowtie= L2)$$

possiamo scrivere con

$$\sigma_P(L1) =\bowtie= \sigma_P(L2)$$

solo che P è supportato sia da L1 che da L2.

Ricapilando, possiamo scrivere

$$\sigma_P(FJ_{\mathcal{H}_T}) = FJ_{\mathcal{H}_T^{\sigma_P}}$$

dove

$$\mathcal{H}_T^{\sigma_P} = \{\sigma_P(Li)|Li \in \mathcal{H}_T\}$$

Con tale notazione possiamo riscrivere la proprietà 4.20 come

$$\boxed{\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}[(FJ_{\mathcal{H}_T^{\sigma_p}})]}$$

- **Caso B2)** Se $\mathcal{H}_\mathcal{T} \neq \emptyset$, e $(\pounds_T - \mathcal{H}_\mathcal{T})$ non T-completo, e $\pounds_\mathcal{A} \not\subseteq \mathcal{H}_\mathcal{T}$

  Supponiamo di avere una interrogazione del tipo $\Pi_\mathcal{A}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T})]$, poniamo per semplicità $P = P_\alpha \wedge P_\beta$. Consideriamo l'insieme $\pounds_T = \{L1,L2,L3\}$ che rappresenta le classi da elaborare per questa query. $\mathcal{H}_\mathcal{T} = \{L1\}$ è un sottoinsieme di $\pounds_T$, e abbiamo $\pounds_\mathcal{A} = \{L1,L3\}$. $(\pounds_T - \mathcal{H}_\mathcal{T}) = \{L2,L3\}$ supportano solo P_α. Nel seguito mostreremo che:

$$\begin{aligned}
\Pi_\mathcal{A}[\sigma_P(FJ_{\pounds_T})] &= \Pi_\mathcal{A}\left[\sigma_P(\underbrace{L1}_{\mathcal{H}_\mathcal{T}} =\bowtie= (\underbrace{L2 =\bowtie= L3}_{\pounds_T - \mathcal{H}_\mathcal{T}})\right] \\
&= \Pi_\mathcal{A}\left[\underbrace{\sigma_{P_\alpha \wedge P_\beta}(L1)}_{\mathcal{H}_\mathcal{T}} =\bowtie= \underbrace{\sigma_{P_\alpha}(L2 =\bowtie= L3)}_{\pounds_T - \mathcal{H}_\mathcal{T}}\right] \\
&= \Pi_\mathcal{A}\left[FJ_{\mathcal{H}_T^{\sigma_p}} =\bowtie \sigma_{P_\alpha}(L3)\right]
\end{aligned}$$

ottenendo quindi la seguente proprietà:

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}\left[FJ_{\mathcal{H}_T^{\sigma_P}} =\bowtie \sigma_{P_\alpha}(L3)\right]$$

Dimostrazione

Consideriamo $\pounds_T = \{L1,L2,L3\}$, l'insieme delle classi locali coinvolte in una query del tipo:

$$\Pi_{\mathcal{A}}[\sigma_P(L2 =\bowtie= L3 =\bowtie= L1)]$$

Abbiamo i seguenti insiemi:

$\pounds_T = \{L1,L2,L3\}$, $\pounds_{\mathcal{A}} = \{L1,L3\}$, $\mathcal{H}_T = \{L1\}$.

Le condizioni seguenti sono verificate:

1. $\mathcal{H}_T \neq \emptyset$
2. $(\pounds_T - \mathcal{H}_T)$ è non T-completo

   ovvero possiamo individuare P_α e P_β tale che $P = P_\alpha \wedge P_\beta$

   $$e \quad \sigma_{P_\alpha}(L2 =\bowtie= L3) \neq \emptyset$$

3. $\pounds_{\mathcal{A}} \not\subseteq \mathcal{H}_T$

$(\pounds_T - \mathcal{H}_T) = \{L2,L3\}$ è non T-completo $\Rightarrow \sigma_{P_\alpha \wedge P_\beta}(L2 =\bowtie= L3) = \emptyset$

(i)

$L1 \in \mathcal{H}_T \Rightarrow \sigma_{P_\alpha \wedge P_\beta}(L1) \neq \emptyset$

(ii)

(i) e (ii) implicano la seguente semplificazione:

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(L2 =\bowtie= L3 =\bowtie= L1)] = \Pi_{\mathcal{A}}[\sigma_{P_\alpha}(L2 =\bowtie= L3) \bowtie= \sigma_{P_\alpha \wedge P_\beta}(L1)]$$

(iii)

$L2 \notin \pounds_{\mathcal{A}}$, $L3 \in \pounds_{\mathcal{A}}$, $\Rightarrow \Pi_{\mathcal{A}}(L2) = \emptyset$ e $\Pi_{\mathcal{A}}(L3) \neq \emptyset$ allora:

$\Pi_{\mathcal{A}}(\sigma_{P_\alpha}(L2 =\bowtie= L3)) = \Pi_{\mathcal{A}}(\sigma_{P_\alpha}(L3)) \neq \emptyset$

(iv)

(iv) implica che:

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha}(L2 =\bowtie= L3) \bowtie= \sigma_{P_\alpha \wedge P_\beta}(L1)] = \Pi_{\mathcal{A}}[\sigma_{P_\alpha}(L3) \bowtie= \sigma_{P_\alpha \wedge P_\beta}(L1)]$$

(v)

Quindi abbiamo verificato che

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}[\sigma_{P_\alpha}(L3) \bowtie= \underbrace{\sigma_{P_\alpha \wedge P_\beta}(L1)}_{\mathcal{H}_T^{\sigma_p}}]$$

cioè

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}[\sigma_{P_\alpha}(L3) \bowtie= FJ_{\mathcal{H}_T^{\sigma_p}}]$$

Esempio 4.3.4

Consideriamo che la seguente query:
$\Pi_{\{C,D,E\}}[\sigma_{P_\alpha \wedge P_\beta}(L1 =\bowtie= L2 =\bowtie= L3)]$ con $S(P_\alpha) = A$ e $S(P_\beta) = B$

Questa query equivale a:

```
select C, D, E
from G
where A op const1 and B op const2
```

Consideriamo che le classi coinvolte sono le stesse di prima :

L1(A,B,C,D,W,K), L2(A,F,V,Y,K), L3(A,F,E,Y,W,K) . Quindi

$\pounds_T = \{L1,L2,L3\}$, $\pounds_{\mathcal{A}} = \{L1,L3\}$, $\mathcal{H}_T = \{L1\}$

$L1$

A	B	C	D	K
a_1	b_1	c_1	d_1	k_1
a_5	b_5	c_5	d_5	k_5

$L2$

A	K
a_1	k_1
a_3	k_3

$L3$

A	E	K
a_3	e_3	k_3
a_5	e_5	k_5
a_7	e_7	k_7

Prima di tutto vediamo cosa otteniamo con la full disjunction nella versione naive:

$$\sigma_{P_\alpha \wedge P_\beta}(L1 =\bowtie= L2 =\bowtie= L3)$$

L1.A	L1.B	L1.C	L1.D	L1.K	L2.A	L2.K	L3.A	L3.E	L3.K	
a_1	b_1	c_1	d_1	k_1	a_1	k_1	$\perp$	$\perp$	$\perp$	$\surd$
a_5	b_5	c_5	d_5	k_5	$\perp$	$\perp$	a_5	e_5	k_5	$\surd$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	a_3	k_3	a_3	e_3	k_3	
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	a_7	e_7	k_7	

Le righe con $\surd$ sono le tuple che costituiscono la risposta alla query.
È facile verificare che lo stesso risultato si ottiene con l'espressione semplificata (ottenuta in precedenza):

$$\Pi_{\{C,D,E\}}[(\sigma_{P_\alpha \wedge P_\beta}(L1) =\bowtie \sigma_{P_\alpha}(L3))]$$

L1.A	L1.B	L1.C	L1.D	L1.K	L3.A	L3.E	L3.K
a_1	b_1	c_1	d_1	k_1	$\perp$	$\perp$	$\perp$
a_5	b_5	c_5	d_5	k_5	a_5	e_5	k_5

Osservazione 4.3.3 *Facendo riferimento alla* 4.3.2*, risulta che gli attributi A e B della classe L1 e l'attributo A di L3 non servono nel calcolo della risposta alla query. Quanto detto nella* 4.3.2 *rimane valido anche in questo caso. Si ricorda che nel caso di più di una classe gli attributi di join vanno conservati. In sostanza si riducono gli schemi delle classi ai soli attributi della proiezione.*

L'applicazione di tale regola permette di ottenere:

L1.C	L1.D	L1.K	L3.E	L3.K
c_1	d_1	k_1	$\perp$	$\perp$
c_5	d_5	k_5	e_5	k_5

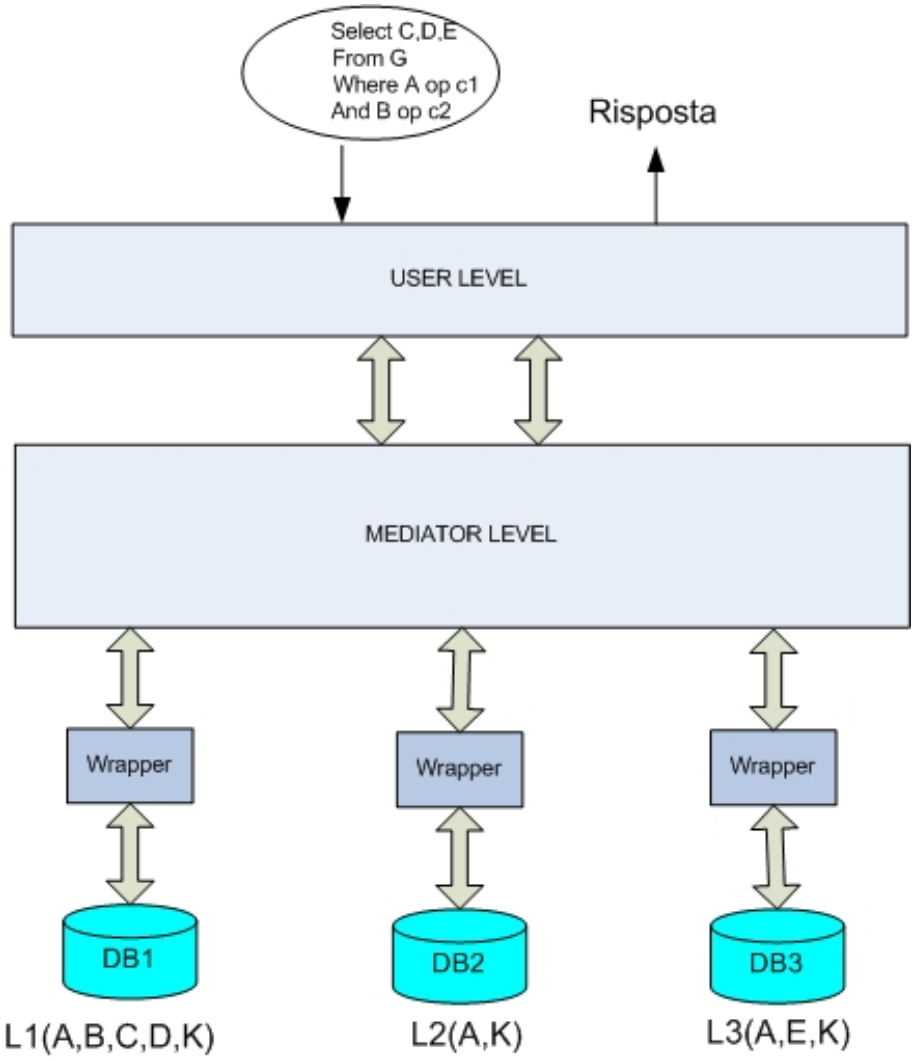

Figura 4.2. Elaborazione della risposta per $\Pi_{\{C,D,E\}}[\sigma_P(FJ_{\mathcal{L}_T})]$

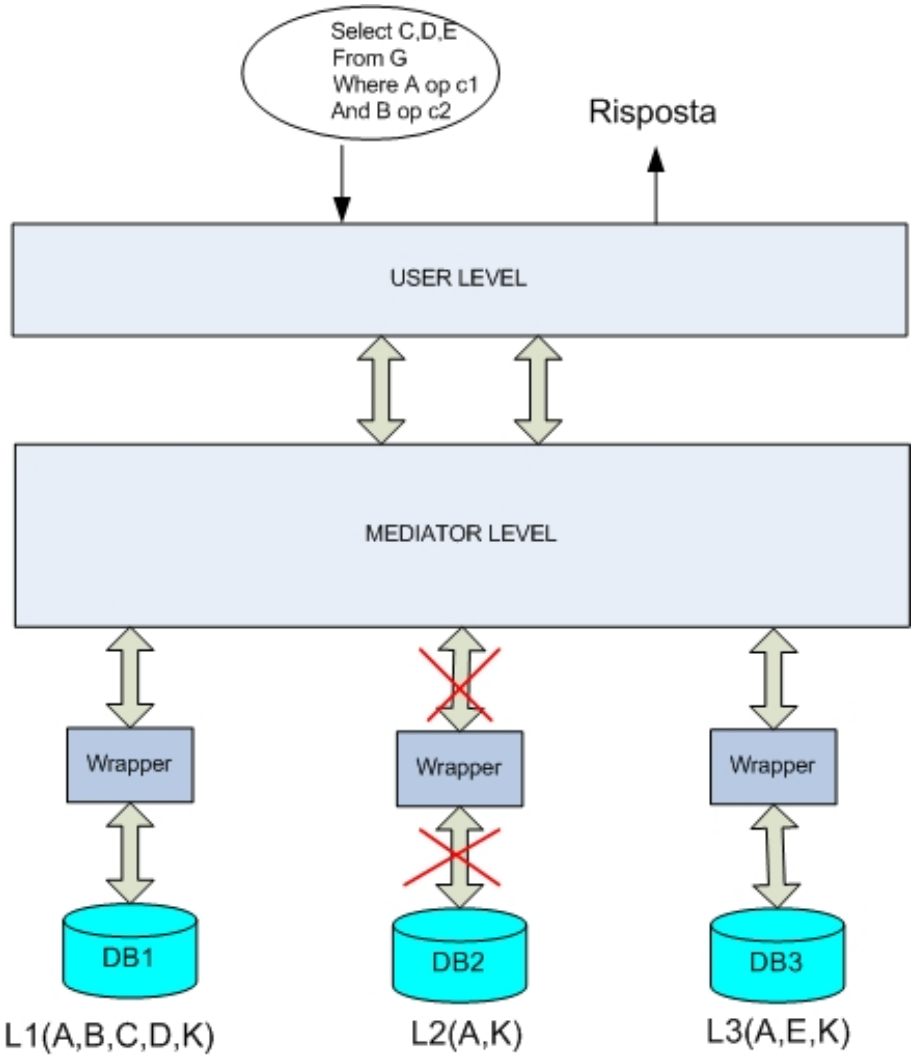

Figura 4.3. Elaborazione della risposta: $\Pi_{\{C,D,E\}}[(FJ_{\pounds_{\mathcal{H}_T^{\sigma_p}}})]$

Proprietà generale nel caso di n classi.
Allo scopo di descrivere in modo più chiaro la proprietà generale introduciamo la seguente 'notazione'.
Consideriamo

$$P = P_\alpha \wedge P_\beta$$

e

$$\sigma_{P_\alpha}(L3)$$

Formalmente sarebbe un errore scrivere

$$\sigma_P(L3)$$

perché L3 non ha tutti gli attributi per supportare P. Se supponiamo che P sia nella forma normale congiuntiva

$$P = P_1 \wedge P_2 \wedge \cdots \wedge P_n$$

allora nel seguito scriveremo

$$\sigma_{\overline{P}}(L)$$

dove $\overline{P}$ è P nel quale ogni P_i non supportato in L è sostituito con $P_i = true$: Notare che $S(\overline{P}) \subset S(P)$.

Con queste premesse la semplificazione ottenuta nell'esempio 4.3.4 viene riscritta come

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}[\sigma_P(L1) =\bowtie \sigma_{\overline{P}}(L3)]$$

ovvero

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}[FJ_{\mathcal{H}_T^{\sigma_p}} =\bowtie \sigma_{\overline{P}}(L3)] \qquad (3)$$

Proprietà nel caso generale

Si parte dalla proprietà (3), considerando

$$card(\mathcal{H}_T) > 1$$

$$\sigma_{\overline{P}}(L3)$$

Cioè dobbiamo considerare che L3 sia corrispondente ad un insieme di classi ovvero che

$$card(\pounds_{\mathcal{A}} - \mathcal{H}_T) > 1$$

quindi nel caso generale

$$\sigma_{\overline{P}}(L3)$$

considerando

$$\pounds_{\mathcal{A}} - \mathcal{H}_T = \{L3,...,LQ\} \qquad Q > 3$$

diventa:

$$\underbrace{\sigma_{\overline{P}}(L3) =\bowtie= \cdots =\bowtie= \sigma_{\overline{P}}(L_Q)}_{(\pounds_{\mathcal{A}} - \mathcal{H}_T)}$$

pertanto

$$\Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T})\right] \;=\; \Pi_{\mathcal{A}}\left[FJ_{\mathcal{H}_T^{\sigma_p}} =\bowtie (\sigma_{\overline{P}}(L3) =\bowtie= \cdots =\bowtie= \sigma_{\overline{P}}(L_Q))\right] \tag{4.21}$$

Sulla base dell'equivalenza 4.5, si scrive la 4.21 come:

$$\Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T})\right] \;=\; \Pi_{\mathcal{A}}\left[\left(\left(FJ_{\mathcal{H}_T^{\sigma_p}} =\bowtie \sigma_{\overline{P}}(L3)\right) =\bowtie \cdots\right) =\bowtie \sigma_{\overline{P}}(L_Q)\right]$$

La proprietà generale ottenuta è:

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}\left[\left(\left(FJ_{\mathcal{H}_T^{\sigma_p}} =\bowtie \sigma_{\overline{P}}(L3)\right) =\bowtie \cdots\right) =\bowtie \sigma_{\overline{P}}(L_Q)\right]$$

Osservazione 4.3.4 *Come detto prima, il calcolo del filtro globale F_r è semplice e quindi finora non è stato considerato. In particolare, nel caso in cui $\pounds_T - \mathcal{H}_T$ è non T-completo risulta essere $F_r = true$, cioè non occorre applicare nessuna condizione dopo il calcolo di FJ. Questa considerazione implica che gli attributi relativi alla condizione di where vanno eliminati dagli schemi delle classi locali coinvolte nella query considerata. Nell'esempio* 4.3.5*, l'eliminazione degli attributi A,B da S(L1) e A da S(L3) non cambia il risultato ottenuto.*

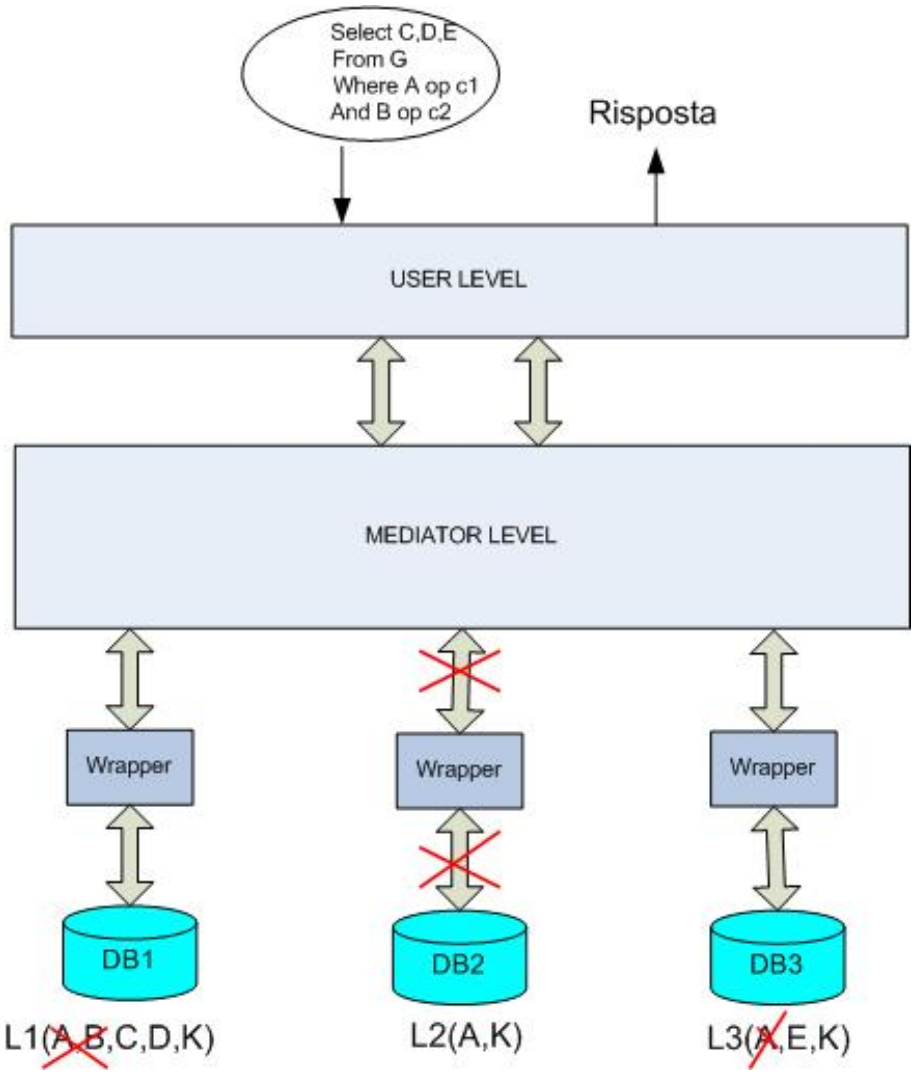

Figura 4.4. Riduzione schemi ai soli attributi di $\mathcal{A}$ e Join

Caso B3) $(\pounds_T - \mathcal{H}_T)$ è T-completo, e $\pounds_\mathcal{A} \subseteq \pounds_T$.

Noi consideriamo che le classi coinvolte in una query sempre della forma $\Pi_\mathcal{A}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T})]$ sono elementi dell'insieme seguente $\pounds_T = \{L1,L2,L3\}$, come nel caso precedente poniamo $P = P_\alpha \wedge P_\beta$. Poi consideriamo $\mathcal{H}_\mathcal{T} = \{L3\}$ e $\pounds_\mathcal{A} = \{L1\}$. E consideriamo che $(\pounds_T - \mathcal{H}_\mathcal{T})$ supportano $P_\alpha \wedge P_\beta$, e L1 supporta P_α, allora facciamo la partizione rispetto ai predicati P_α e P_β:
$\pounds_{P_\alpha} = \{L1,L3\}$, e $\pounds_{P_\beta} = \{L2\}$ dimostriamo che:

$$\begin{aligned}
\Pi_\mathcal{A}[\sigma_P(FJ_{\pounds_T})] &= \Pi_\mathcal{A}\left[\sigma_{P_\alpha \wedge P_\beta}(\underbrace{L1 =\bowtie= L3}_{\pounds_{P_\alpha}} =\bowtie= \underbrace{L2}_{\pounds_{P_\beta}})\right] \\
&= \Pi_\mathcal{A}\left[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_{P_\alpha}}) =\bowtie \sigma_{P_\beta}(FJ_{\pounds_{P_\beta}})\right] \\
&= \Pi_\mathcal{A}\left[\sigma_P(FJ_{\pounds_{P_\alpha}}) =\bowtie \sigma_{\overline{P}}(L2)\right]
\end{aligned}$$

ottenendo quindi la seguente proprietà:

$$\boxed{\Pi_\mathcal{A}[\sigma_P(FJ_{\pounds_T})] = \Pi_\mathcal{A}\left[\sigma_P(FJ_{\pounds_{P_\alpha}}) =\bowtie \sigma_{\overline{P}}(L2)\right]}$$

Dimostrazione

1. $\pounds_\mathcal{A} \subseteq \pounds_T$

2. $\mathcal{H}_T \neq \emptyset$

3. $(\pounds_T - \mathcal{H}_T)$ è T-completo

   ovvero possiamo individuare P_α e P_β tale che $P = P_\alpha \wedge P_\beta$

$$\sigma_P(L1 =\bowtie= L2) \neq \emptyset \quad ma \quad \sigma_P(L1) = \sigma_P(L2) = \emptyset$$

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(L1 =\bowtie= L3 =\bowtie= L2)] = \Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T})]$$

L'espressione $\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T})$ è scritta come segue:

$$\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T}) = (\sigma_{P_\alpha \wedge P_\beta}(L3) =\bowtie= \sigma_{P_\beta}(L2) =\bowtie= \sigma_{P_\alpha}(L1))$$

$\pounds_T = \{L1,L3,L2\}$, $\mathcal{H}_T = \{\ L3\ \}$, $\pounds_{\mathcal{A}} = \{L1\}$

$(\pounds_T - \mathcal{H}_T) = \{L1,L2\}$ è T-completo $\Rightarrow \sigma_{P_\alpha \wedge P_\beta}(L1 =\bowtie= L2) \neq \emptyset$

(i)

$L3 \in \mathcal{H}_T \Rightarrow \sigma_{P_\alpha \wedge P_\beta}(L3) \neq \emptyset$

(ii)

(i) e (ii) implicano che:

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(L1 =\bowtie= L3 =\bowtie= L2)] = \Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(L1 =\bowtie= L2) =\bowtie= \sigma_{P_\alpha \wedge P_\beta}(L3)]$$

Questa equivalenza non mi porta a nessuna trasformazione cioè non si può semplificare nessun full outerjoin come nei casi discussi precedentemente. Per cui bisogna ricercare un'altra strategia in modo da riuscire a semplificare l'espressione.

Facciamo la partizione di $\pounds_T$ considerando $\pounds_{P_\alpha}$ per primo, e otteniamo i seguenti insiemi:

$\pounds_{P_\alpha} = \{L1,L3\}$, $\pounds_{P_\beta} = \{L2\}$, $\mathcal{H}_T = \{L1\}$

Per L2$\in \pounds_{P_\beta}$, $\sigma_{P_\alpha \wedge P_\beta}(L2) = \emptyset$

(iii)

L1,L3 $\in \pounds_{P_\alpha}$ di conseguenza $\mathcal{H}_T \subseteq \pounds_{P_\alpha} \Rightarrow \sigma_{P_\alpha \wedge P_\beta}(L1 =\bowtie= L3) \neq \emptyset$

(iv)

(iii) e (iv) implicano la riscrittura di $\sigma_P(FJ_{\pounds_T})$ con $P = P_\alpha \wedge P_\beta$ in modo semplificato:

$$\begin{aligned}
\sigma_P(FJ_{\pounds_T}) &= \sigma_{P_\alpha \wedge P_\beta}((L3 =\bowtie= L2) =\bowtie= L1)) \\
&= \sigma_{P_\alpha \wedge P_\beta}(\underbrace{L3 =\bowtie= L1}_{\pounds_{P_\alpha}}) =\bowtie= \sigma_{P_\beta}(\underbrace{L2}_{P_\beta}) \\
&= \sigma_P(FJ_{\pounds_{P_\alpha}}) =\bowtie= \sigma_{\overline{P}} FJ_{\pounds_{P_\beta}} \\
&= \sigma_P(FJ_{\pounds_{P_\alpha}}) =\bowtie \sigma_{\overline{P}}(FJ_{\pounds_{P_\beta}}) \\
&= \sigma_P(FJ_{\pounds_{P_\alpha}}) =\bowtie \sigma_{\overline{P}}(\underbrace{L2}_{P_\beta})
\end{aligned}$$

Esempio 4.3.5

Consideriamo la seguente query

```
select C, D
from G
where W op const1 and Y op const2
```

Con $S(P_\alpha) = W$ e $S(P_\beta) = Y$, e $P = P_\alpha \wedge P_\beta$

Le classi coinvolte sono le stesse dei casi precedenti:

L1(A,B,C,D,W,K), L2(A,F,V,Y,K) e L3(A,E,W,Y,K)

$$\Pi_{\mathcal{A}}[\sigma_P(L1 =\bowtie= L2 =\bowtie= L3)]$$

$\pounds_T = \{L1,L2,L3\}$, $\pounds_{\mathcal{A}} = \{L1\}$ e $\mathcal{H}_T = \{L3\}$

Consideriamo la seguente partizione P_α e P_β:

$\pounds_{P_\alpha} = \{L1,L3\}$ e $\pounds_{P_\beta} = \{L2\}$ quindi abbiamo:

$$\begin{aligned}
\Pi_{\mathcal{A}}[\sigma_P(L1 =\bowtie= L2 =\bowtie= L3)] &= \Pi_{\mathcal{A}}[\sigma_P((L1 =\bowtie= L3) =\bowtie L2)] \\
&= \Pi_{\mathcal{A}}[\sigma_P((L1 =\bowtie= L3) =\bowtie L2)]
\end{aligned}$$

Consideriamo le seguenti tabelle:

$L1$

C	D	W	K
c_1	d_1	w_1	k_1
c_3	d_3	w_3	k_3
c_5	d_5	w_5	k_5

$L2$

Y	K
y_1	k_1
y_3	k_3
y_4	k_4

$L3$

W	Y	K
w_1	y_1	k_1
w_5	y_5	k_5
w_7	y_7	k_7

A questo punto calcoliamo la $FJ_{\pounds_T}$ nella sua versione naive:

$$\sigma_P(L1 =\bowtie= L2 =\bowtie= L3)$$

L1.C	L1.D	L1.W	L1.K	L2.Y	L2.K	L3.W	L3.Y	L3.K	
c_1	d_1	w_1	k_1	y_1	k_1	w_1	y_1	k_1	$\surd$
c_3	$\perp$	w_3	k_3	y_3	k_3	w_3	y_3	k_3	$\surd$
c_5	$\perp$	w_5	k_5	$\perp$	$\perp$	w_5	y_5	k_5	$\surd$
$\perp$	$\perp$		$\perp$	$\perp$	y_4	k_4 $\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	w_7	y_7	k_7	$\surd$

Le tuple con $\surd$ fanno parte della riposta della query.

Applichiamo la nostra semplificazione:

$$\sigma_P((L1 =\bowtie= L3) =\bowtie L2)$$

L1.C	L1.D	L1.W	L1.K	L3.W	L3.Y	L3.K	L2.Y	L2.K	
c_1	d_1	w_1	k_1	w_1	y_1	k_1	y_1	k_1	$\surd$
c_3	$\perp$	w_3	k_3	w_3	y_3	k_3	y_3	k_3	$\surd$
c_5	$\perp$	w_5	k_5	w_5	y_5	k_5	$\perp$	$\perp$	$\surd$
$\perp$	$\perp$	$\perp$	$\perp$	w_7	y_7	k_7	$\perp$	$\perp$	$\surd$

In questa tabella si può notare come le tuple che fanno parte della risposta sono sempre le stesse come prima ma con una tupla in meno che è quella irrilevante. Bisogna notare che non sempre questa semplificazione permette di ottenere un'ottimizzazione: per esempio se avessimo considerato invece la seguente sequenza $P_\beta = \{L2,L3\}$ e $P_\alpha = \{L1\}$ nella partizione, avremo ottenuto la seguente semplificazione:

$$\sigma_P(L1 \bowtie= (L3 =\bowtie= L2)$$

Ecco la tabella che otteniamo:

$$\sigma_P(L1 =\bowtie= L2 =\bowtie= L3)$$

L1.C	L1.D	L1.W	L1.K	L2.Y	L2.K	L3.W	L3.Y	L3.K	
c_1	d_1	w_1	k_1	y_1	k_1	w_1	y_1	k_1	$\surd$
c_3	$\perp$	w_3	k_3	y_3	k_3	w_3	y_3	k_3	$\surd$
c_5	$\perp$	w_5	k_5	$\perp$	$\perp$	w_5	y_5	k_5	$\surd$
$\perp$	$\perp$		$\perp$	$\perp$	y_4	k_4 $\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	w_7	y_7	k_7	$\surd$

Questa tabella è identica alla prima calcolata nella versione naive. Evidentemente se la tabella L1 avesse delle tuple che non fanno match con $(L2 =\bowtie= L3)$, quelle tuple veranno eliminate. In certo senso possiamo dire che la nostra semplificazione nel caso peggiore ricada nella versione naive (non semplificata).

Proprietà generale nel caso di n classi: Consideriamo i seguenti insiemi:

$\pounds_T = \{L1,...,Ln\}$, facciamo la partizione come abbiamo visto nella nostra definizione considerando per primo: $\pounds_{P_\alpha} = \{L1,...,L_K\}$ e $\pounds_{P_\beta} = \{L_{K+1},...,Ln\}$. Visto che $\mathcal{H}_T$ non è vuoto allora in questa partizione abbiamo $\pounds_{P_\alpha}$ già T-completo visto che le classi di $\mathcal{H}_T$ sono contenute in quel sottoinsieme. La semplificazione generale è quella seguente:

$$\begin{aligned}
\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] &= \Pi_{\mathcal{A}}\left[\sigma_P((\underbrace{L1 =\bowtie= \cdots =\bowtie= L_K}_{\pounds_{P_\alpha}}) =\bowtie= (\underbrace{L_{K+1} =\bowtie= \cdots =\bowtie= Ln}_{\pounds_{P_\beta}}))\right] \\
&= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_{P_\alpha}} =\bowtie= FJ_{\pounds_{P_\beta}})\right] \\
&= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_{P_\alpha}}) =\bowtie \sigma_{\overline{P}}(FJ_{\pounds_{P_\beta}})\right] \\
&= \Pi_{\mathcal{A}}\left[\left(\left(\sigma_P(FJ_{\pounds_{P_\alpha}}) =\bowtie \sigma_{\overline{P}}(L_{K+1})\right) =\bowtie \cdots\right) =\bowtie \sigma_{\overline{P}}(Ln)\right]
\end{aligned}$$

La proprietà generale ottenuta è:

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T})] = \Pi_{\mathcal{A}}\left[\left(\left(\sigma_P(FJ_{\pounds_{P_\alpha}}) =\bowtie \sigma_{\overline{P}}(L_{K+1})\right) =\bowtie \cdots\right) =\bowtie \sigma_{\overline{P}}(Ln)\right]$$

Caso 2: $\pounds_{\mathcal{A}} \not\subseteq \pounds_T$

Nel caso 2, si deve considerare anche $\pounds_{\mathcal{A}}$.

Pertanto

$$\pounds = \pounds_T \cup \pounds_A$$

Si ricorda inoltre che le query con un solo predicato possono godere delle proprietà ricavate in questo caso.

Caso D) Se $\pounds_{\mathcal{A}} \cap \pounds_T = \emptyset$

consideriamo i seguenti insieme riferiti sempre al contesto della nostra query specificata all'inizio di questo capitolo con i predicati $P_\alpha \wedge P_\beta$.

$\pounds_T = \{L1, L3\}$, e $\pounds_{\mathcal{A}} = \{L2\}$ e quindi con

$$\pounds_T \cap \pounds_{\mathcal{A}} = \emptyset$$

Se poniamo $P = P_\alpha \wedge P_\beta$ allora si può scrivere:

$$\begin{aligned}
\Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds})\right] &= \Pi_{\mathcal{A}}\left[\sigma_P((\underbrace{L1 =\!\bowtie\!= L3}_{\pounds_T}) =\!\bowtie\!= \underbrace{L2}_{\pounds_{\mathcal{A}}})\right] \\
&= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) =\!\bowtie\!= \sigma_{true}(FJ_{\pounds_{\mathcal{A}}})\right] && (4.22)\\
&= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) =\!\bowtie \sigma_{true}(L2)\right] && (4.23)\\
&= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) \bowtie \sigma_{true}(FJ_{\pounds_{\mathcal{A}}})\right] && (4.24)
\end{aligned}$$

La proprietà 4.23 è ottenuta applicando la regola del null rejection: $FJ_{\pounds_T}$ è l'unico operando che soddisfa il predicato P per cui il full outerjoin della 4.22 viene sostituito dal left outerjoin. La 4.24 è ottenuta per il fatto che $FJ_{\pounds_{\mathcal{A}}}$ è l'unico operando che contiene gli attributi della proiezione. Visto che le classi coinvolte in $FJ_{\pounds_T}$ non hanno attributi ($\pounds_T \cap \pounds_{\mathcal{A}} = \emptyset$) di proiezione, di conseguenza $FJ_{\pounds_T}$ ha solo tuple nulle sugli attributi della proiezione. Avendo all'inizio posto tutti gli attributi not null ($P_{\mathcal{A}}$), l'applicazione di tale condizione implica la sostituizione del left outerjoin dal join. Pertanto la proprietà ottenuta nel caso D1 è:

$$\Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds})\right] = \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) \bowtie \sigma_{true}(FJ_{\pounds_{\mathcal{A}}})\right]$$

Dimostrazione

Consideriamo la seguente query:

$$\Pi_{\mathcal{A}}[\sigma_P(L1 =\bowtie= L2 =\bowtie= L3)]$$

e i seguenti insiemi: $\pounds_{\mathcal{A}} = \{L2\}$ e $\pounds_T = \{L1,L3\}$

$\pounds_T$ è sempre T-completo $\Rightarrow \sigma_P(L1 =\bowtie= L3) \neq \emptyset$

(i)

$L2 \notin \pounds_T \Rightarrow \sigma_P(L2) = \emptyset$

(ii)

(i) e (ii) implicano la seguente semplificazione:

$$\Pi_{\mathcal{A}}[\sigma_P(L1 =\bowtie= L3 =\bowtie= L2)] = \Pi_{\mathcal{A}}[\sigma_P(\underbrace{(L1 =\bowtie= L3)}_{\pounds_T} =\bowtie \underbrace{L2}_{\pounds_{\mathcal{A}}})] \qquad (iii)$$

$\pounds_T \cap \pounds_{\mathcal{A}} = \emptyset \Rightarrow \Pi_{\mathcal{A}}(L1 =\bowtie= L3) = \emptyset$, e $L2 \in \pounds_{\mathcal{A}} \Rightarrow \Pi_{\mathcal{A}}(L2) \neq \emptyset$ (iv)

(iv) implica che la seguente semplificazione:

$$\Pi_{\mathcal{A}}[\sigma_P(\underbrace{(L1 =\bowtie= L3)}_{\pounds_T} =\bowtie \underbrace{L2}_{\pounds_{\mathcal{A}}} = \Pi_{\mathcal{A}}[\sigma_P(\underbrace{(L1 =\bowtie= L3)}_{\pounds_T} \bowtie \underbrace{L2}_{\pounds_{\mathcal{A}}})]$$

abbiamo dimostrato:

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})] = [\sigma_P(FJ_{\pounds_T}) \bowtie \sigma_{true}(FJ_{\pounds_{\mathcal{A}}})]$$

Per il calcolo di $FJ_{\pounds_T}$ basta guardare le proprietà del caso 1 e vedere in quale caso ricade l'insieme $\pounds_T$.

Esempio 4.3.6

Consideriamo la seguente query:

select F, V
from G
where Y op const1

Abbiamo $S(P_\alpha) = \{Y\}$, $\Pi_{\mathcal{A}} = \{F,V\}$ e poniamo $P = P_\alpha$. Le classi coinvolte sono le seguenti:

L1(A,B,C,D,W,K), L2(A,F,V,Y,K) e L3(A,F,E,Y,W,K)

$\pounds_T = \{L1,L3\}$, e $\pounds_{\mathcal{A}} = \{L2\}$:

Prima di tutto si elabora $\pounds_T$ per ricavare l'espressione di $FJ_{\pounds_T}$. Allora abbiamo:

$\mathcal{H}_T = \pounds_T$ visto che anche $\mathcal{H}_T = \{L1,L3\}$ **(1)**, e si verifica che $(\pounds_T - \mathcal{H}_T)$ è non T-completo. Ricordiamo che $FJ_{\pounds_T}$ può essere semplificato come nel caso 1.

Ma sappiamo che tutte le classi contenute in $\mathcal{H}_T$ vengono coinvolte nel calcolo con l'operatore di full outerjoin, quindi:

$$\sigma_P(FJ_{\pounds_T}) = FJ_{\mathcal{H}_T^{\sigma_P}} = \sigma_P(L1 =\bowtie= L3)$$

$\sigma_P(L1 =\bowtie= L3)$

L1.Y	L1.K	L3.Y	L3.K
y_1	k_1	y_1	k_1
y_4	k_4	$\perp$	$\perp$
$\perp$	$\perp$	y_2	k_2

$L2$

L2.F	L2.V	L2.K
f_1	v_1	k_1
f_6	v_6	k_6
f_4	v_4	k_4
f_5	v_5	k_5

Il calcolo della FJ nella sua versione naive ci dà il seguente risultato:

$$FJ_{\pounds} = \sigma_P(L1 =\bowtie= L3 =\bowtie= L2)$$

L1.Y	L1.K	L3.Y	L3.K	L2.F	L2.V	L2.K	
y_1	k_1	y_1	k_1	f_1	v_1	k_1	$\surd$
y_4	k_4	$\perp$	$\perp$	f_4	v_4	k_4	$\surd$
$\perp$	$\perp$	y_2	k_2	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	$\perp$	f_5	v_5	k_5	
$\perp$	$\perp$	$\perp$	$\perp$	f_6	v_6	k_6	

Le tuple in $\surd$ sono quelle che fanno parte del risultato. Notiamo che la terza riga della tabella ottenuta soddisfa la condizione di WHERE ma non può visualizzare nessun valore degli attributi presenti nella SELECT. Quindi risulta irrelevante per la nostra query perché non fornisce le informazioni richieste. La quarta e la quinta riga non servono perché non soddisfa il primo requisito: quello di verificare la condizione della clausola WHERE.Per poter eliminare le tuple di questo tipo faremo uso della 4.24. Per cui risulta meglio riscrivere la nostra $FJ_{\pounds}$ come segue:

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})] = \Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds_T}) \bowtie \sigma_{true}(FJ_{\pounds_{\mathcal{A}}})]$$

L'applicazione di questa semplificazione ci permette di ottenere il seguente risultato:

$$\sigma_P(FJ_{\pounds_T}) \bowtie \sigma_{true}(FJ_{\pounds_{\mathcal{A}}})$$

L1.Y	L1.K	L3.Y	L3.K	L2.F	L2.V	L2.K
y_1	k_1	y_1	k_1	f_1	v_1	k_1
y_4	y_4	$\perp$	$\perp$	f_4	v_4	k_4

Nella versione ottimizzata vediamo quanto le tuple irrelevanti vengono eliminate direttamente con l'operatore di join. Il numero di tuple ottenute risulta molto inferiore a quello della tabella precedente. Ecco quello ottenuto dopo proiezione sugli attributi {F,V}:

$$\Pi_{\{F,V\}}[\sigma_P(FJ_{\pounds_T}) \bowtie (FJ_{\pounds_{\mathcal{A}}})]$$

L2.F	*L2.V*
f_1	v_1
f_4	v_4

Proprietà generale: Consideriamo $card(\pounds_{\mathcal{A}}) \geq 1$

$$\pounds = \{L1,...,Ln\}, \qquad \pounds_{\mathcal{A}} = \{Lm,...,Ln\}$$

$$\begin{aligned}
\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})] &= \Pi_{\mathcal{A}}\left[(\sigma_P((FJ_{\pounds_T}) =\!\bowtie\!= \underbrace{\sigma_{true}(Lm =\!\bowtie\!= \cdots =\!\bowtie\!= Ln)}_{\pounds_{\mathcal{A}}})\right] \\
&= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) =\!\bowtie\!= \sigma_{true}(FJ_{\pounds_{\mathcal{A}}})\right] \\
&= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) =\!\bowtie \sigma_{true}(FJ_{\pounds_{\mathcal{A}}})\right] \\
&= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) \bowtie \sigma_{true}(FJ_{\pounds_{\mathcal{A}}})\right]
\end{aligned}$$

La proprietà generale ottenuta con $card(\pounds_{\mathcal{A}}) > 1$, è quella seguente:

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})] = \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) \bowtie \sigma_{true}(FJ_{\pounds_{\mathcal{A}}})\right]$$

Si nota che questa formula non è differente da quella iniziale salvo che si considera $card(\pounds_{\mathcal{A}}) > 1$ visto che all'inizio abbiamo considerato $card(\pounds_{\mathcal{A}}) = 1$.

Si osserva come nella 4.5, che si può anche scrivere la formula generale ottenuta come segue:

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})] = \Pi_{\mathcal{A}}\left[(((\sigma_P(FJ_{\pounds_T})) =\bowtie \sigma_{true}(Lm)) =\bowtie \cdots) =\bowtie \sigma_{true}(Ln)\right]$$

La seconda formula ottenuta non mi garantisce più la condizione $P_{\mathcal{A}}$, in quanto non sempre tutte le tuple di $FJ_{\pounds_T}$ saranno eliminate ciò dovuto all'assenza di join fra $FJ_{\pounds_T}$ e Lm (si guarda la dimostrazione della 4.5). Per illustrare quanto detto,vediamo il seguente esempio.

```
select C, E
from G
where V op const1
```

$L1(A,B,C,D,W,K)$, L2(A,F,V,Y,K) e L3(A,E,W,Y,K)

$P = P_\alpha = \{V\}$, $\Pi_{\mathcal{A}} = \{C,E\}$, abbiamo i seguenti insiemi:

$\pounds_T = \{L2\}$, $\pounds_{\mathcal{A}} = \{L1,L3\}$, $\pounds = \{L1,L2,L3\}$

$L2$

V	K
v_1	k_1
v_3	k_3
v_5	k_5

$L1$

C	K
c_1	k_1
c_3	k_3
c_4	k_4

$L3$

E	K
e_1	k_1
e_2	k_2
e_3	k_3

abbiamo:

$$\sigma_P(FJ_{\pounds}) = \sigma_P(L2 =\bowtie= L1 =\bowtie= L3)$$

Ecco la tabella che otteniamo con la versione naive di $FJ_{\pounds}$

L2.V	L2.K	L1.C	L1.K	L3.E	L3.K	
v_1	k_1	c_1	k_1	e_1	k_1	$\surd$
v_3	k_3	c_3	k_3	e_3	k_3	$\surd$
v_5	k_5	$\perp$	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	c_4	k_4	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	$\perp$	e_2	k_2	

La prima semplificazione $FJ_{\mathcal{L}} = \sigma_P(L2 \bowtie (L1 =\bowtie= L3)) = \sigma_P(FJ_{\mathcal{L}_T}) \bowtie \sigma_{true} FJ_{\mathcal{L}_\mathcal{A}})$ viene applicata e otteniamo:

L2.V	L2.K	L1.C	L1.K	L3.E	L3.K
v_1	k_1	c_1	k_1	e_1	k_1
v_3	k_3	c_3	k_3	e_3	k_3

La seconda ottimizzazione è $FJ_{\mathcal{L}} = \sigma_P(FJ_{\mathcal{L}_T}) =\bowtie \sigma_{true}(L1)) =\bowtie \sigma_{true}(L3))$

L2.V	L2.K	L1.C	L1.K	L3.E	L3.K	
v_1	k_1	c_1	k_1	e_1	k_1	$\surd$
v_3	k_3	c_3	k_3	e_3	k_3	$\surd$
v_5	k_5	$\perp$	$\perp$	$\perp$	$\perp$	

La differenza fra le due semplificazioni si trova soprattutto nei risultati intermedi del calcolo della $FJ_{\mathcal{L}}$ e non tutte le tuple di $FJ_{\mathcal{L}_T}$ sono state eliminate per la seconda ottimizzazione. Ma riteniamo che le due semplificazioni risultano migliori della versione naive di $FJ_{\mathcal{L}}$.

Caso C) $\mathcal{L}_\mathcal{A} \not\subseteq \mathcal{L}_T$, e $\mathcal{L}_\mathcal{A} \cap \mathcal{L}_T \neq \emptyset$

In questo caso si considera che $\mathcal{L}_T \cap \mathcal{L}_\mathcal{A} \neq \emptyset$, (per semplicità scriveremo $\mathcal{L}_\mathcal{A} - \mathcal{L}_\mathcal{A} \cap \mathcal{L}_T$ come $\mathcal{L}_\mathcal{A} - \mathcal{L}_T$).
Essendo

$$\mathcal{L} = \mathcal{L}_T \cup \mathcal{L}_A$$

abbiamo che $(X = \mathcal{L}_\mathcal{A} - \mathcal{L}_T)$

$$FJ_{\mathcal{L}} = FJ_{\mathcal{L}_T} =\bowtie= FJ_X$$

allora, intuitivamente, la semplificazione di $FJ_{\mathcal{L}}$ avverrà semplificando $FJ_{\mathcal{L}_T}$ come già fatto nel caso 1 e semplificando FJ_X.

Consideriamo i seguenti insiemi che descrivono il contesto della nostra query:

$\pounds_{\mathcal{A}} = \{L1,L3,L2\}$, e $\pounds_T = \{L2\}$, si può notare come

$$\pounds_{\mathcal{A}} \not\subseteq \pounds_T \quad e \quad \pounds_{\mathcal{A}} \cap \pounds_T \neq \emptyset$$

per cui si può scrivere:

$$\begin{aligned}
\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})] &= \Pi_{\mathcal{A}}\left[\sigma_P((\underbrace{L2}_{\pounds_T}) =\bowtie= (\underbrace{L1 =\bowtie= L3}_{X}))\right] && (4.25)\\
&= \Pi_{\mathcal{A}}\left[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T}) =\bowtie \sigma_{true}(FJ_X)\right] && (4.26)\\
&= \Pi_{\mathcal{A}}\left[\left(\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T}) =\bowtie \sigma_{true}(L3)\right) =\bowtie \sigma_{true}(L1)\right] && (4.27)
\end{aligned}$$

ciò ci porta a capire che la computazione di $FJ_{\pounds_T}$ mi danno delle tuple che mi visualizzano valori non nulli sugli attributi della proiezione per cui il left outerjoin fra $FJ_{\pounds_T}$ e $FJ_{\pounds_T - \pounds_{\mathcal{A}}}$ non può essere sostituito dal join perchè si perderebbe alcune tuple di $FJ_{\pounds_T}$ che hanno valori non nulli sugli attributi della proiezione. Quindi dopo l'applicazione della proprietà di null rejection nella 4.25 si sostituisce il full outerjoin da left outerjoin nella 4.26. Ottenuto ciò, non si può fare altre semplicazioni in più. Però riteniamo la 4.27 più ottimo per il fatto che dopo l'elaborazione della $FJ_{\pounds_T}$, tutte le classi di $(\pounds_{\mathcal{A}} - \pounds_T)$ vengono coinvolte con l'operatore di left outerjoin invece della 4.26 in cui prima tutte le classi di $(\pounds_{\mathcal{A}} - \pounds_T)$ vengono coinvolte con l'operatore di full outerjoin nell'espressione $FJ_{\pounds_{\mathcal{A}} - \pounds_T}$ per poi applicare l'operatore di left outerjoin al risultato intermedio di $FJ_{\pounds_{\mathcal{A}} - \pounds_T}$ con $FJ_{\pounds_T}$.

La proprietà ottenuta è:

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})] = \Pi_{\mathcal{A}}\left[\left(\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T}) =\bowtie \sigma_{true}(L3)\right) =\bowtie \sigma_{true}(L1)\right]$$

Dimostrazione

Consideriamo la seguente query:

$$\Pi_{\mathcal{A}}[\sigma_P(L1 =\bowtie= L3 =\bowtie= L2)]$$

Consideriamo i seguenti insiemi:

$\pounds_T = \{L2\}$, $\pounds_{\mathcal{A}} = \{L1, L3, L2\}$

$\pounds_T = \{L2\}$ è T-completo $\Rightarrow \sigma_P(L2) \neq \emptyset$ (i)

$X = \{L1, L3\} \Rightarrow \sigma_P(L1 =\bowtie= L2) = \emptyset$ (ii)

allora (i) e (ii) implicano la seguente semplificazione:

$$\Pi_{\mathcal{A}}[\sigma_P(L1 =\bowtie= L3 =\bowtie= L2)] = \Pi_{\mathcal{A}}[\sigma_P(\underbrace{L2}_{\pounds_T} =\bowtie (\underbrace{L2 =\bowtie= L3}_{X}))]$$

L'espressione può essere riscritta come nella 4.5, per cui otteniamo:

$$\Pi_{\mathcal{A}}[\sigma_P(L2 =\bowtie (L1 =\bowtie= L3))] = \Pi_{\mathcal{A}}[\sigma_P((\underbrace{L2}_{\pounds_T} =\bowtie L1) =\bowtie L3)]$$

Esempio 4.3.7

```
select C, E, Y
from G
where V op const1
```

Abbiamo le seguenti classi: L1(A,B,C,D,W,K), L2(A,F,V,Y,K) e L3(A,E,W,Y,K)

Dall'analisi della query otteniamo i seguenti insiemi:

$\pounds_T = \{L2\}$, e $\pounds_{\mathcal{A}} = \{L1, L3, L2\}$

abbiamo le seguenti tabelle:

$L2$

V	Y	K
v_1	y_1	k_1
v_3	$\perp$	k_3
v_5	y_5	k_5

$L1$

C	K
c_1	k_1
c_4	k_4

$L3$

E	Y	K
e_1	y_1	k_1
e_2	y_2	k_2
e_3	y_3	k_3

Ecco la tabella che otteniamo con la versione naive di $FJ_{\pounds}$

L2.V	L2.Y	L2.K	L1.C	L1.K	L3.E	L3.Y	L3.K	
v_1	y_1	k_1	c_1	k_1	e_1	y_1	k_1	$\surd$
v_3	$\perp$	k_3	$\perp$	$\perp$	e_3	y_3	k_3	$\surd$
v_5	y_5	k_5	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\surd$
$\perp$	$\perp$	$\perp$	c_4	k_4	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	e_2	y_2	k_2	

Mentre l'applicazione della nostra ottimizzazione ci permette di ottenere il risultato seguente:

$$\Pi_{\mathcal{A}}[\sigma_P((L2 =\bowtie L1) =\bowtie L3)] \quad = \quad \Pi_{\mathcal{A}}[(\sigma_P(FJ_{\pounds_T}) =\bowtie \sigma_{true}(L1)) =\bowtie \sigma_{true}(L3)]$$

L2.V	L2.Y	L2.K	L1.C	L1.K	L3.E	L3.Y	L3.K	
v_1	y_1	k_1	c_1	k_1	e_1	y_1	k_1	$\surd$
v_3	$\perp$	k_3	$\perp$	$\perp$	e_3	y_3	k_3	$\surd$
v_5	y_5	k_5	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\surd$

Le tuple della riga 4 e 5 sono state eliminate. La cardinalità delle tuple del risultato finale è determinata dalla cardinalità delle tuple del calcolo dell'espressione di $FJ_{\pounds_T}$. l'espressione di $FJ_{\pounds_T}$ può assumere qualunque delle espressioni ottenute al caso 1 salvo il caso B2.

Proprietà generale: Consideriamo il seguente insieme:

$X = \{Lm,...,Ln\}$ con $P = P_\alpha \wedge P_\beta$, allora si può scrivere:

$$\begin{aligned}
\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})] &= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) =\bowtie= \sigma_{true}(\underbrace{Lm =\bowtie= \cdots =\bowtie= Ln}_{X})\right] \\
&= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) =\bowtie= \sigma_{true}(FJ_X)\right] \\
&= \Pi_{\mathcal{A}}\left[\sigma_P(FJ_{\pounds_T}) =\bowtie \sigma_{true}(FJ_X)\right] \\
&= \Pi_{\mathcal{A}}\left[((\sigma_P(FJ_{\pounds_T} =\bowtie \sigma_{true}(Lm)) =\bowtie \cdots) =\bowtie \sigma_{true}(Ln)\right]
\end{aligned}$$

La proprietà generale per il caso C è:

$$\Pi_{\mathcal{A}}[\sigma_P(FJ_{\pounds})] = \Pi_{\mathcal{A}}\left[((\sigma_P(FJ_{\pounds_T} =\bowtie \sigma_{true}(Lm)) =\bowtie \cdots) =\bowtie \sigma_{true}(Ln)\right]$$

Osservazione 4.3.5 *Bisognar ricordare che nel caso* 2*, le proprietà ottenute possono essere applicate anche ad una query avendo un solo predicato. La sola differenza sta sull'espressione di* $FJ_{\pounds_T}$ *che rappresenta l'elaborazione delle classi di* $\pounds_T$*.*
Nel caso di un solo predicato, P *è supportato da tutte le classi di* $\pounds_T$*, allora tutte le classi di* $\pounds_T$ *sono le classi di* $\mathcal{H}_T$ *cioè* $\pounds_T = \mathcal{H}_T$*. Le classi di* $\pounds_T$ *verranno coinvolte con un full outerjoin perché* $\pounds_T = \mathcal{H}_T$ *(vedi caso B1 e B2) quindi* $FJ_{\pounds_T}$ *viene riscritta come* $FJ_{\mathcal{H}_T^{\sigma_P}}$*.*
Considerando che $\pounds_T \cap \pounds_{\Pi_\mathcal{A}} \neq \emptyset$*. Allora in modo generale avremo:*

$$\begin{aligned}
\Pi_\mathcal{A}[\sigma_P(FJ_\pounds)] &= \Pi_\mathcal{A}\left[((FJ_{\pounds_T} =\!\bowtie \sigma_{true}(Lm)) =\!\bowtie \cdots) =\!\bowtie \sigma_{true}(Ln)\right] \\
&= \Pi_\mathcal{A}\left[FJ_{\mathcal{H}_T^{\sigma_P}} =\!\bowtie \sigma_{true}(FJ_X)\right] \\
&= \Pi_\mathcal{A}\left[\left(\left(FJ_{\mathcal{H}_T^{\sigma_P}} =\!\bowtie \sigma_{true}(Lm)\right)\cdots\right) =\!\bowtie \sigma_{true}(Ln)\right]
\end{aligned}$$

Proprietà ottenuta per un solo predicato e con le seguenti condizioni:

$$\pounds_\mathcal{A} \not\subseteq \pounds_T \qquad e \qquad \pounds_\mathcal{A} \cap \pounds_T \neq \emptyset$$

$$\boxed{\Pi_\mathcal{A}[\sigma_P(FJ_\pounds)] = \Pi_\mathcal{A}\left[\left(\left(FJ_{\mathcal{H}_T^{\sigma_P}} =\!\bowtie \sigma_{true}(Lm)\right)\cdots\right) =\!\bowtie \sigma_{true}(Ln)\right]}$$

Questa proprietà ha l'obiettivo di attirare la nostra attenzione sulla situazaione in cui c'è un solo predicato. Evidentemente quando abbiamo un solo predicato si può verificare che $\pounds_T = \mathcal{H}_\mathcal{T}$ *e si può notare come ricadiamo nei casi B1 e B2 perché abbiamo* $\pounds_T - \mathcal{H}_\mathcal{T} = \emptyset$ *è non T-completo (vedi definizione* 4.2.4*).*
Però nel caso di un predicato, la semplificazione è interessante solo nel caso in cui abbiamo $\pounds_\mathcal{A} \not\subseteq \pounds_T$ *perché se avessimo* $\pounds_\mathcal{A} \subseteq \pounds_T$ *allora ci avrebbe* $\Pi_\mathcal{A}[FJ_\pounds] = \Pi_\mathcal{A}[FJ_{\pounds_T}]$ *visto che (*$\pounds_\mathcal{A} \subseteq \pounds_T$ *e* $\pounds = \pounds_T \cup \pounds_\mathcal{A}$*. Sapendo che* $\pounds_T = \mathcal{H}_\mathcal{T}$ *perché abbiamo un solo predicato allora si avrebbe:*

$$\Pi_\mathcal{A}[FJ_{\pounds_T}] = \Pi_\mathcal{A}[FJ_{\mathcal{H}_T^{\sigma_P}}]$$

Quindi tutte le classi coinvolte nella query saranno coinvolte nella computazione di FJ con l'operatore di full outerjoin. Questo risultato ci fa capire che non abbiamo ottenuto miglioramenti.Di conseguenza nel caso di un predicato, sarà preso in considerazione solo il caso in cui $\pounds_{\mathcal{A}} \not\subseteq \pounds_T$.

Riconsideriamo la query sottoposta all'inizio del nostro studio:

Cercare l'indirizzo di tutti gli automobilisti di 26 anni che hanno una macchina di tipo Fiat.

```
select Nome, Cognome, Città, Via
from Automobilisti
where TipoAuto='Fiat' and Età='26'
```

$S(P_\alpha) = \{Fiat\}$, $P_\beta = \{Età\}$, $T = S(P_1) \cup S(P_2) = \{TipoAuto,Età\}$
$\mathcal{A} = \{Nome,Cognome,Città,Via\}$. l'insieme delle classi coinvolte nella query:

$$\pounds = \pounds_T \cup \pounds_{\mathcal{A}} = \{ANAGRAFE,ASSICURATI,PATENTE\}$$

con

$$\pounds_T = \{ANAGRAFE,ASSICURATI\}, \quad \pounds_{\mathcal{A}} = \{ANAGRAFE,PATENTE\}$$

Si calcola l'insieme X:

$$X = \pounds_{\mathcal{A}} - \pounds_T = \{ASSICURATI\}$$

Si verifica che

$$\pounds_{\mathcal{A}} \not\subseteq \pounds_T \quad e \quad \pounds_{\mathcal{A}} \cap \pounds_T \neq \emptyset$$

Per cui applichiamo il caso C:

$$\begin{aligned} \Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds})] &= \Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T}) =\!\bowtie \sigma_{true}(FJ_X)] \\ &= \Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T}) =\!\bowtie \sigma_{true}(PATENTE)] \end{aligned}$$

A questo punto semplifichiamo l'espressione di $FJ_{\pounds_T}$.
Si verifica che

$$\mathcal{H}_T = \emptyset$$

Per cui applichiamo il caso A:

$$\begin{aligned}\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds_T}) &= \sigma_{P_\alpha}(FJ_{\pounds_{P_\alpha}}) \bowtie \sigma_{P_\beta}(FJ_{\pounds_{P_\beta}}) \\ &= \sigma_{P_\alpha}(ANAGRAFE) \bowtie \sigma_{P_\beta}(ASSICURATI)\end{aligned}$$

Infine l'espressione di $FJ_{\pounds}$ è:

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds})] = \Pi_{\mathcal{A}}[(\sigma_{P_\alpha}(ANAGRAFE) \bowtie \sigma_{P_\beta}(ASSICURATI)) =\bowtie \sigma_{true}(PATENTE)]$$

Le istanze di ANAGRAFE (L1), PATENTE (L2), ASSICURATI (L3) sono:

ANAGRAFE (L1)

Nome	Cognome	Età	Città	Via
Mario	Verde	26	Vignola	Araldi 61
Fabrizio	Sala	26	Rubiera	Vittorio 13
Ugo	Po	26	Carpi	Pighi 18
Cristina	Valter	26	Modena	campi 29

ASSICURATI (L3)

Nome	Cognome	Marca
Mario	Verde	Fiat
Ugo	Po	Fiat
Roberto	Galla	Fiat

PATENTE (L2)

Nome	Cognome
Mario	Verde
Ugo	Po
Riberto	Galla
Stefi	Grina

L'applicazione delle regole di semplificazione sull'operazione di Full Join ci dà il seguente risultato:

$$\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds}) = (\sigma_{P_\alpha}(ANAGRAFE) \bowtie \sigma_{P_\beta}(ASSICURATI)) =\bowtie \sigma_{true}(PATENTE)$$

L1.Nome	L1.Cognome	L1.Età	L1.Città	L1.Via	L3.Nome	L3.Cognome	L3.Marca	L2.Nome	L2.Cognome
Mario	Verde	26	Vignola	Araldi 61	Mario	Verde	Fiat	Mario	Verde
Ugo	Po	26	Carpi	Pighi 18	Ugo	Po	Fiat	Ugo	Po

Si osserva che lo stesso risultato si poteva ottenere verificando la minimalità di $\pounds_T$ (si guarda la 4.2.3).
Di conseguenza si elimina rispettivamente gli attributi Età dallo schema di ANAGRAFE e Marca dallo schema di ASSICURATI (si guarda la 4.3.1). Per cui otteniamo la stessa semplificazione come prima e in più meno attributi coinvolti nel calcolo della risposta alla query, con $ANAGRAFE'$(Nome,Cognome,Città,Via) e $ASSICURATI'$(Nome,Cognome).

$$\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds}) = (\sigma_{P_\alpha}(ANAGRAFE') \bowtie \sigma_{P_\beta}(ASSICURATI')) =\bowtie \sigma_{true}(PATENTE)$$

L1.Nome	L1.Cognome	L1.Città	L1.Via	L3.Nome	L3.Cognome	L2.Nome	L2.Cognome
Mario	Verde	Vignola	Araldi 61	Mario	Verde	Mario	Verde
Ugo	Po	Carpi	Pighi 18	Ugo	Po	Ugo	Po

Mentre senza le nostre regole di semplificazione, otteniamo:

$$FD(\sigma_{P_\alpha}(ANAGRAFE),\sigma_{P_\beta}(ASSICURATI)),\sigma_{true}(PATENTE))$$
$$= \sigma_{P_\alpha}(ANAGRAFE) =\bowtie= \sigma_{P_\beta}(ASSICURATI) =\bowtie= \sigma_{true}(PATENTE)$$

$$\sigma_{P_\alpha}(ANAGRAFE) =\bowtie= \sigma_{P_\beta}(ASSICURATI) =\bowtie= \sigma_{true}(PATENTE)$$

L1.Nome	L1.Cognome	L1.Età	L1.Città	L1.Via	L3.Nome	L3.Cognome	L3.Marca	L2.Nome	L2.Cognome	
Mario	Verde	26	Vignola	Araldi 61	Mario	Verde	Fiat	Mario	Verde	√
Ugo	Po	26	Carpi	Pighi 18	Ugo	Po	Fiat	Ugo	Po	√
Fabrizio	Sala	26	Rubiera	Vittorio 13	⊥	⊥	⊥	⊥	⊥	
Cristina	Valter	26	Modena	campi 29	⊥	⊥	⊥	⊥	⊥	
⊥	⊥	⊥	⊥	⊥	Roberto	Galla	Fiat	Roberto	Galla	
⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	Stefi	Grina	

La risposta della query é:

$$\Pi_{\{Cognome,Nome,Città,Via\}}[\sigma_{P_\alpha \wedge P_\beta}(FJ_{\pounds})]$$

Nome	Cognome	Città	Via
Mario	Verde	Vignola	Araldi 61
Ugo	Po	Carpi	Pighi 18

Con questa illustrazione pratica, abbiamo voluto mettere in luce una situazione reale in cui le proprietà ottenute nel nostro studio possono essere usate per ridurre sia il tempo di risposta della query e che sia la quantità di risorse impiegate. Inoltre, le trasformazioni ottenute possono godere di una ulteriore ottimizzazione attraverso l'uso degli algoritmi esistenti per l'elaborazione di operazione di join oppure di Left (right) outerjoin.

Capitolo 5

Riduzione degli Schemi delle Classi in una Query

In questo capitolo discuteremo la possibilità di effettuare il 'push' della proiezione sulle classi locali (per diminuire la cosidetta arità della relazione risultato, ovvero il numero di attributi che costituiscono la relazione risultato); tale operazione comporta una diminuzione della cardinalità della risposta locale e di conseguenza una ottimizzazione dell'interrogazione. Come abbiamo già detto all'inizio del capitolo precedente, per una migliore presentazione dei risultati raggiunti nella tesi, prima abbiamo presentato equivalenze algebriche che consentono sia di eliminare alcuni termini coinvolti nell'espressione di full outerjoin sia la possibilità di sostituire l'operazione di full outerjoin con un'operazione di left outerjoin, di right outerjoin oppure di join. In questo capitolo ci occuperemo anche della semplificazione degli schemi delle classi locali coinvolti nelle query. In questo modo saremo in grado di definire un unico metodo (descritto tramite un algoritmo) che combina le due possibilità: eliminare delle classi oppure semplificarne lo schema.

5.1 Il concetto di riduzione degli schemi

Nell'esecuzione di una query che coinvolge più di una classe, spesso accade che due o più classi hanno gli attributi locali mappati sugli stessi attributi globali. Nel contesto dell'omogeneita semantica, ciò potrebbe portare ad una certa ridondanza dovuta al fatto che attributi globali simili per classi locali implicano valori uguali. La nostra idea è quella di trovare un modo di porter eliminare quelli attributi ripetuti in altre classi e conservandoli in altre in modo da evitare di avere molte colonne . Però bisognar individuare in quale contesto possiamo applicare il nostro concetto di riduzione di schemi delle classi locali. In questa sezione, noi presenteremo un'esempio di query e dopo faremo notare la nostra intuizione sull'ottimizzazione della query

elaborata. Prima di tutto noi consideriamo le query sotto la forma già specificata nel capitolo precedente cioè query che hanno la condizione di selezione con questa forma: $P_\alpha \wedge P_\beta \cdots \wedge P_\gamma$. La cardinalità di T ($card(T)$), è il numero di predicati della condizione di selezione. Il nostro studio presenta due casi, casi da cui cercheremo di formalizzare il nostro algoritmo.

Caso 1: $card(T) = 1$ e $\pounds_{\mathcal{A}} \not\subseteq \pounds_T$

Per illustrare il nostro approccio, abbiamo la seguente query:

Esempio 5.1.1

select F, Y
from G
where E op const1

Consideriamo le stesse classi locali che sono state definite dalla nostra mapping table alla pagina 49.

$$L1(A,B,C,D,W,K),\ \ L2(A,F,V,Y,K),\ \ L3(A,E,W,Y,K)$$

$\pounds_T = \{L3\}$, $\pounds_{\mathcal{A}} = \{L2,L3\}$, $S(P_\alpha) = \{E\}$ e $\pounds_{P_\alpha} = \{L3\}$.

Consideriamo le seguenti tabelle:

L2

F	Y	K
f_1	y_1	k_1
f_2	y_2	k_2
f_3	y_3	k_3
f_4	y_4	k_4
f_5	$\perp$	k_5

L3

E	Y	K
e_1	y_1	k_1
e_2	y_2	k_2
e_5	$\perp$	k_5

Calcoliamo il full outerjoin di queste classi per ottenere la risposta alla query:

$$\sigma_{true}(L2) =\bowtie= \sigma_{P_\alpha}(L3)$$

L2.F	L2.Y	L2.K	L3.E	L3.Y	L3.K	
f_1	y_1	k_1	e_1	y_1	k_1	$\surd$
f_2	y_2	k_2	e_2	y_2	k_2	$\surd$
f_3	y_3	k_3	$\perp$	$\perp$	$\perp$	
f_4	y_4	k_4	$\perp$	$\perp$	$\perp$	
f_5	$\perp$	k_5	e_5	$\perp$	k_5	$\surd$

L'applicazione del filtro sul risultato del full outerjoin ci dà il seguente risultato:

$$\sigma_{F_r}(\sigma_{true}(L2) =\bowtie= \sigma_{P_\alpha}(L3))$$

L2.Y	L2.F	L2.K	L3.Y	L3.E	L3.K
y_1	f_1	k_1	y_1	e_1	k_1
y_2	f_2	k_2	y_2	e_2	k_2
$\perp$	f_5	k_5	$\perp$	e_5	k_5

Risultato della query

Dopo la compattazione delle colonne che hanno lo stesso attributo globale e la proiezione otteniamo:

$$\Pi_{\{F,Y\}}[\sigma_{F_r}(\sigma_{true}(L2) =\bowtie= \sigma_{P_\alpha}(L3))]$$

F	Y
f_1	y_1
f_2	y_2
f_5	$\perp$

Possiamo notare come le colonne L2.Y e L3.Y della tabella ottenuta dopo l'applicazione del filtro presentano gli stessi valori sulle tuple che fanno match: ciò non ci sorprende visto che siamo in condizione di omogeneità semantica.Di conseguenza si può pensare di eliminare una delle due colonne nel calcolo della full disjunction. Facciamo uso degli insiemi che abbiamo definito nel capitolo precedente. Per la query sottoposta nell'esempio 5.1.1, abbiamo i seguenti insiemi:

$\pounds_T = \{L3\}$, $\pounds_{\mathcal{A}} = \{L3,L2\}$, $\pounds_{P_\alpha} = \{L3\}$ con $S(P_\alpha) = \{E\}$, e $\Pi_{\mathcal{A}} = \{F,Y\}$

Il nostro approccio è il seguente:

1. Prima si considera il numero di predicati nella where, nel nostro caso essendo un solo predicato non c'è bisogno di fare la partizione di $\pounds_T$.
   Nel nostro caso possiamo scrivere:
   $$\pounds_T = \pounds_{P_\alpha} = \{L3(Y,E,K)\}\ ;$$
2. si considera l'insieme $\pounds'_{\mathcal{A}} = \pounds_{\mathcal{A}} - (\pounds_{\mathcal{A}} \cap \pounds_T)$.

   Nel nostro esempio abbiamo:
   $$\pounds'_{\mathcal{A}} = \{L2\}$$

3. Poi consideriamo l'insieme degli attributi della proiezione $\mathcal{A}$.
   Nel nostro esempio abbiamo:
   $$\mathcal{A} = \{F,Y\}$$

4. Preso un attributo della $\mathcal{A}$, si verifica se tutti gli schemi delle classi di $\pounds_T$ lo possiedono;

5. Se vero, allora si riducono gli schemi delle classi di $\pounds'_{\mathcal{A}}$ che hanno l'attributo considerato e si riparte dal passo 4;
   Nel nostro caso abbiamo:
   $$\pounds'_{\mathcal{A}} = \pounds_{\mathcal{A}} - (\pounds_{\mathcal{A}} \cap \pounds_T) = L2(F,Y,K)$$

6. L'algoritmo si ferma quando tutti gli attributi di $\mathcal{A}$ sono stati esaminati.

L'applicazione di questi passi ci porta a ridurre lo schema della classe L2(F,Y,K) al nuovo schema L2(F,K): visto che tutti gli schemi delle classi di $\pounds_T$ possiedono l'attributo Y: nel nostro esempio abbiamo solo la classe L3(Y,E,K).

$$L2(F, Y, K) \overset{riduzione\ schema}{\longrightarrow} L2(F, K)$$

Riprendiamo il calcolo del full outerjoin delle istanze delle classi L2 e L3 con lo schema di L2 ridotto.

L2

F	K
f_1	k_1
f_2	k_2
f_3	k_3
f_4	k_4
f_5	k_5

L'istanza di L3 considerata è quella precedentemente utilizzata nel primo calcolo del full outerjoin.

$$\sigma_{true}(L2) =\bowtie= \sigma_{P_\alpha}(L3)$$

L2.F	L2.K	L3.E	L3.Y	L3.K	
f_1	k_1	e_1	y_1	k_1	$\surd$
f_2	k_2	e_2	y_2	k_2	$\surd$
f_3	k_3	$\perp$	$\perp$	$\perp$	
f_4	k_4	$\perp$	$\perp$	$\perp$	
f_5	k_5	e_5	$\perp$	k_5	$\surd$

risultato del full outerjoin di L1 e L2 con suo schema ridotto

Il risultato dopo l'operazione del filtro residuo globale è:

$$\sigma_{F_r}(\sigma_{true}(L2) =\bowtie= \sigma_{P_\alpha}(L3))$$

L2.F	L2.K	L3.Y	L3.E	L3.K
f_1	k_1	Y_1	e_1	k_1
f_2	k_2	Y_2	e_2	k_2
f_5	k_5	$\perp$	e_5	k_5

Risultato della query con lo schema ridotto di L2

Dopo compattazione delle colonne, otteniamo lo stesso risultato come prima però con lo schema ridotto di L2.

$$\Pi_{\{F,Y\}}[\sigma_{F_r}(\sigma_{true}(L2) =\bowtie= \sigma_{P_\alpha}(L3))]$$

F	Y
f_1	y_1
f_2	y_2
f_5	$\perp$

Noi concludiamo questo caso, faccendo notare che questa riduzione è possibile solo nell'assunzione dell'omogeneità semantica. *È* stato descritto in modo semplice la metodologia di riduzione degli schemi delle classi coinvolte nella query. Quelle classi che possono subire una possibile riduzione di schema sono classi dell'insieme:
$\pounds'_{\mathcal{A}} = \pounds_{\mathcal{A}} - (\pounds_{\mathcal{A}} \cap \pounds_T)$. Nel caso di un solo predicato, la riduzione è applicabile solo se abbiamo $\pounds_{\mathcal{A}} \not\subseteq \pounds_T$.

Caso 2: $card(T) > 1$

In questo caso noi consideriamo query con più di un predicato. Non viene considerato la condizione $\pounds_{\mathcal{A}} \not\subseteq \pounds_T$ perché la metodologia descritta risolve sia quel caso che il caso $\pounds_{\mathcal{A}} \subseteq \pounds_T$.Questo è possibile perchè le classi che andremo a ridurre solo quelle apparteneti all'insieme '$(\pounds \cap \pounds_{P_i}) - \pounds_{P_i}$' con $\pounds = \pounds_T \cup \pounds_{\mathcal{A}}$. Consideriamo la seguente query:

Esempio 5.1.2

```
select A, Y, V
from G
where B op const1 and E op const2
```

Abbiamo

$$S(P_\alpha) = \{E\} \quad e \quad S(P_\beta) = \{B\}$$

Consideriamo le stesse classi locali che sono state definite dalla nostra mapping table alla pagina 49.

L1(A,B,C,D,W,K), L2(A,F,V,Y,K), L3(A,E,W,Y,K)

Per cui abbiamo

$$\pounds_T = \{L1,L3\} \quad e \quad \pounds_{\mathcal{A}} = \{L1,L2,L3\}$$

poi si calcola:

$$\pounds'_{\mathcal{A}} = \pounds_{\mathcal{A}} - (\pounds_{\mathcal{A}} \cap \pounds_T) = \{L2\} \quad e \quad \pounds = \pounds_T \cup \pounds_{\mathcal{A}} = \{L1,L2,L3\}$$

Le classi che possono subire una riduzione di schemi appartengono all'insieme:

$$(\pounds - (\pounds \cap \pounds_{P_i}))$$

, si può notare come in questo caso non è più necessario calcolare $\pounds'_{\mathcal{A}}$ perché è già contenuto in $(\pounds - (\pounds \cap \pounds_{P_i}))$. Se $\pounds'_{\mathcal{A}}$ è vuoto è sempre contenuto in quell'insieme appena citato ; $\pounds'_{\mathcal{A}}$ vuoto implica che $\pounds_{\mathcal{A}} \subseteq \pounds_T$.

Ecco le istanze delle classi coinvolte nella nostra query:

L1

A	B	K
a_1	b_1	k_1
a_2	b_2	k_2
a_4	b_4	k_4

L2

A	Y	V	K
a_1	y_1	v_1	k_1
a_3	$\perp$	$\perp$	k_3
a_5	y_5	v_5	k_5

L3

A	Y	E	K
a_1	y_1	e_1	k_1
a_2	y_2	e_2	k_2
a_6	y_6	e_6	k_6

Dopo il calcolo della Full disjunction otteniamo:

$$(\sigma_{P_\alpha}(L1) =\bowtie= \sigma_{true}(L2) =\bowtie= \sigma_{P_\beta}(L3))$$

L1.A	L1.B	L1.K	L2.A	L2.Y	L2.V	L2.K	L3.A	L3.Y	L3.E	L3.K	
a_1	b_1	k_1	a_1	y_1	v_1	k_1	a_1	y_1	e_1	k_1	√
a_2	b_2	k_2	$\perp$	$\perp$	$\perp$	$\perp$	a_2	y_2	e_2	k_2	√
a_4	b_4	k_4	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	a_3	$\perp$	$\perp$	k_3	$\perp$	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	a_5	y_5	v_5	k_5	$\perp$	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	a_6	y_6	e_6	k_6	

Il risultato finale della nostra query è il seguente:

$$\Pi_{\{A,Y,V\}}[\sigma_{P_\alpha}(L1) =\bowtie= \sigma_{true}(L2) =\bowtie= \sigma_{P_\beta}(L3)]$$

A	Y	V
a_1	y_1	v_1
a_2	y_2	$\perp$

Nelle seguenti righe proponiamo un algoritmo che analizza la nostra query come finora è stato fatto basandosi sugli attributi dei predicati, sugli attributi della proiezione e sugli schemi delle classi coinvolte nella query.

Il nostro approccio è il seguente:

1. Prima si considera il numero di predicati nella where, nel nostro caso essendo maggiore di uno, allora faremo la partizione di $\pounds_T$ rispetto ai predicati.
   Nel nostro esempio abbiamo:
   $$\pounds_{P_\alpha} = \{L1(A,B,K)\} \quad e \quad \pounds_{P_\beta} = \{L3(A,E,Y,K)\}$$
2. si considera l'insieme $\pounds = \pounds_{\mathcal{A}} \cup \pounds_T = \{L2,L1,L3\}$;
3. Poi consideriamo l'insieme degli attributi della proiezione $\mathcal{A}$.
   Nel nostro esempio abbiamo:
   $$\mathcal{A} = \{A,Y,V\}$$
4. Preso un attributo della $\mathcal{A}$, si verifica se tutti gli schemi delle classi di $\pounds_{P_i}$ lo possiedono;

5. Se vero, si sceglie l'insieme $\pounds_{P_i}$ che ha soddisfatto il passo 4 per primo; allora si riduce gli schemi delle classi di

$$[\pounds - (\pounds \cap \pounds_{P_i})]$$

che hanno l'attributo considerato; poi si riparte dal passo 4;

6. L'algoritmo si ferma quando tutti gli attributi di $\Pi_{\mathcal{A}}$ sono stati esaminati.

Nel nostro esempio, abbiamo:

$\mathcal{A} = \{A,Y,V\}$, $\pounds = \{L1(A,B,K); L2(A,Y,V,K); L3(A,E,Y,K)\}$,

$\pounds_{P_\alpha} = \{L1\}$ e $\pounds_{P_\beta} = \{L3\}$

Noi consideriamo l'attributo A di $\mathcal{A}$ (I^o ciclo).
L'insieme $\pounds_{P_\alpha} = L1(A,B,K)$ ha tutte le sue classi che hanno i loro schemi che possiedono l'attributo A
(abbiamo una sola classe L1, il plurale serve per generalizzare).
Allora riduciamo gli schemi delle classi di

$$(\pounds - (\pounds \cap \pounds_{P_\alpha})) = \{L2(A,Y,V,K), L3(A,E,Y,K)\}$$

Allora otteniamo gli schemi di L2 e L3 ridotti di A: L2(Y,V,K) e L3(E,Y,K)
Per cui abbiamo:

$$\pounds = \{L1(A,B,K);\ \ L2(Y,V,K);\ \ L3(E,Y,K),\}$$

aggiornato degli schemi di L2 e L3.
Si riparte dal passo 4.

Si considera Y questa volta (II^o ciclo).
$\pounds_{P_\beta} = \{L3(E,Y,K)\}$ (schema aggiornato) ha tutte le sue classi i cui schemi possiedono l'attributo Y.
Quindi si può ridurre gli schemi delle classi di:

$$(\pounds - (\pounds \cap \pounds_{P_\beta})) = \{L2(Y,V,K),\ L1(A,B,K)\}$$

allora L2 sarà l'unica classe il cui schema sarà ridotto di Y: L2(A,V,K).
Per cui abbiamo:

$$\pounds = \{L1(A,B,K);\ \ L2(V,K);\ \ L3(E,Y,K)\}$$

aggiornato solo lo schema di L2 (ridotto per una seconda volta)

Infine si considera l'attributo V, si nota che nè $\pounds_{P_\alpha}$ nè $\pounds_{P_\beta}$ soddisfanno il passo 4.
Di conseguenza non si fa niente e visto che tutti gli attributi di $\mathcal{A}$ sono stati esaminati: il nostro algoritmo termina.

Il nostro algoritmo ha ridotto lo schema di L2(A,Y,V,K) a L2(V,K) e L3(A,E,Y,K) a L3(E,Y,K).
Osserviamo che potevamo ridurre lo schema di L1(A,B,K) a L1(B,K) però lo schema di L3 non sarebbe più potuto ridotto. E L3 sarebbe rimasto invariabile nel nostro esempio.

$$\left.\begin{array}{l} L2(\,A\,,Y\,,V\,,K\,) \\ \\ L3(\,A\,,E\,,Y\,,K\,) \end{array}\right\} \Longrightarrow riduzione\ \ schemi \Longrightarrow \left\{\begin{array}{l} L2(\,V\,,K\,) \\ \\ L3(\,E\,,Y\,,K\,) \end{array}\right.$$

Ricalcoliamo il risultato della query fatta nella 5.1.2, considerando solo le classi L1,L2 e L3 con i loro schemi ridotti.

$$FD(\sigma_{P_\alpha}(L1),\sigma_{true}(L2),\sigma_{P_\beta}(L3))$$

L1.A	L1.B	L1.K	L2.V	L2.K	L3.Y	L3.E	L3.K	
a_1	b_1	k_1	v_1	k_1	y_1	e_1	k_1	$\surd$
a_2	b_2	k_2	$\perp$	$\perp$	y_2	e_2	k_2	$\surd$
a_4	b_4	k_4	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	$\perp$	k_3	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	v_5	k_5	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	y_6	e_6	k_6	

Risultato finale della query dopo la proiezione:

$$\Pi_{\{A,Y,V\}}[\sigma_{F_r}(\sigma_{P_\alpha}(L1) =\bowtie= \sigma_{true}(L2) =\bowtie= \sigma_{P_\beta}(L3))]$$

A	Y	V
a_1	y_1	v_1
a_2	y_2	$\perp$

A questo punto possiamo generalizzare i risultati ottenuti nei due casi analizzati, riportando nel seguente algoritmo il procedimento per la semplificazione degli schemi.

Algoritmo di semplificazione degli schemi

INPUT: $\pounds_T$, $\pounds_{\mathcal{A}}$, $\mathcal{A}$, T

$\pounds = \pounds_T \cup \pounds_{\mathcal{A}}$

IF $card(\pounds_T) > 1$ oppure $card(\pounds_T) = 1$ e $\pounds_{\mathcal{A}} \not\subseteq \pounds_T$

1. Calcolo delle partizioni di $\pounds_T$ rispetto ai predicati:

$$\pounds_{P_1}, \pounds_{P_2}, \ldots, \pounds_{P_m}$$

2. Per ogni attributo $A \in \mathcal{A}$
   Per ogni $\pounds_{P_i}$
   Se per ogni $L \in \pounds_{P_i}$ si ha che $A \in S(L)$
   allora
   Riduzione schema delle classi in $\pounds - \pounds_{P_i}$ dell'attributo A:

$$S(L) = S(L) - A, \quad \forall L \in (\pounds - \pounds_{P_i})$$

Conclusione

Abbiamo dimostrato come ridurre gli schemi delle classi coinvolte in una query senza cambiare il risultato della nostra query. Si è usata l'ipotesi di omogeneità semantica. Evidentemente questa riduzione è applicabile solo nel caso di un predicato oppure di n predicati sotto la seguente forma $P_\alpha \wedge P_\beta \cdots \wedge P_\gamma$. Sono stati definiti due approcci per analizzare le query e determinare se è possibile ridurre gli schemi delle classi coinvolte in questa query. Il risultato ottenuto è molto interessante in quanto può essere accoppiato alle proprietà di semplicazione ottenute nel capitolo precedente. Illustriamo quanto detto applicando le proprietà di semplificazione sulla query dell'esempio 5.1.2.

Riportiamo la Query posta nell'esempio 5.1.2.

$$\Pi_{\{A,Y,V\}}[\sigma_{\{B\ op\ c1\ \wedge\ E\ op\ c2\}}(FJ_{\{L1,\ L2',\ L3'\}}]$$

Abbiamo visto che le classi iniziali coinvolte nella query sono:

$$L1(A,B,K),\ \ L2(A,Y,V,K)\ \ e\ \ L3(A,E,Y,K)$$

L'applicazione del nostro algoritmo ha ridotto gli schemi di L2 e L3:

$$L2(V,K)\ \ \ e\ \ \ L3(E,Y,K)$$

Poniamo: $L2^{'}$ rappresenta L2 con schema ridotto e $L3^{'}$ per L3

Applichiamo adesso le nostre proprietà, abbiamo:

$$\mathcal{H}_{\mathcal{T}} = \emptyset \qquad\qquad (1)$$

,

$$e\ \ \pounds_{\mathcal{A}} \not\subseteq \pounds_T \quad e \quad \pounds_{\mathcal{A}} \cap \pounds_T \neq \emptyset \qquad (2)$$

La condizione 2 ci indica che la proprietà C1 è quella adatta. Per cui abbiamo:

$$\Pi_{\mathcal{A}}[\sigma_{\{B\ op\ c1\ \wedge\ E\ op\ c2\}}(FJ_{\pounds})] = \Pi_{\{A,Y,V\}}[\sigma_{\{B\ op\ c1\ \wedge\ E\ op\ c2\}}(FJ_{\pounds_T}) =\bowtie \sigma_{true}(FJ_{\pounds_{\mathcal{A}}-\pounds_X})]$$

con $X = \pounds_{\mathcal{A}} - (\pounds_{\mathcal{A}} \cap \pounds_T)$

La condizione 1 ci indica che la proprietà A è quella adatta per $FJ_{\pounds_T}$, per cui scriviamo:

$$\sigma_{\{B\ op\ c1\ \wedge\ E\ op\ c2\}}(FJ_{\pounds_T}) = \sigma_{\{B\ op\ c1\}}(FJ_{\pounds_{P_\alpha}}) \bowtie \sigma_{\{E\ op\ c2\}}(FJ_{\pounds_{P_\beta}})$$

$$\sigma_{\{B\ op\ c1\ \wedge\ E\ op\ c2\}}(FJ_{\pounds_T}) = \sigma_{\{B\ op\ c1\}}(L1) \bowtie \sigma_{\{E\ op\ c2\}}(L2^{'})$$

Per cui otteniamo la semplificazione finale:

$$\Pi_{\{A,Y,V\}}[\sigma_{\{B\ op\ c1\ \wedge\ E\ op\ c2\}}(FJ_{\pounds})] = \Pi_{\{A,Y,V\}}[\sigma_{\{B\ op\ c1\}}(L1) \bowtie \sigma_{\{E\ op\ c2\}}(L2^{'}) =\bowtie \sigma_{true}(L3^{'})]$$

In modo semplice possiamo riportare la semplificazione generale ottenuta combinando l'algoritmo di riduzione degli schemi e le proprietà di semplificazione.

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha}(L1) =\bowtie= \sigma_{P_\beta}(L3) =\bowtie= \sigma_{true}(L2)] \qquad (I)$$

$$\Downarrow$$

$$Algoritmo\ di\ Riduzione\ Schemi\ +\ proprietà\ semplificazione$$

$$\Downarrow$$

$$\Pi_{\mathcal{A}}[\sigma_{P_\alpha}(L1) \bowtie \sigma_{P_\beta}(L3^{'}) =\bowtie \sigma_{true}(L2^{'})] \qquad (II)$$

L'ordine di applicazione è molto importante: si applica l'algoritmo di riduzione degli schemi per primo perché facendo così noi andremo a prelevare nella fase di interrogazione dei singoli siti le istanze delle classi che sono state ottenute dopo il processo di riduzione degli schemi. Si ricorda inoltre che una classe il cui schema è stato ridotto ai soli attributi di Join va eliminata se La classe non appartiene a $\pounds_T$. (ipotesi di full connected). Illustriamo i risultati (I) e (II) considerando le relazioni risultanti da tali espressioni

Risultato (I)

L1.A	L1.B	L1.K	L2.A	L2.Y	L2.V	L2.K	L3.A	L3.Y	L3.E	L3.K	
a_1	b_1	k_1	a_1	y_1	v_1	k_1	a_1	y_1	e_1	k_1	$\surd$
a_2	b_2	k_2	$\perp$	$\perp$	$\perp$	$\perp$	a_2	y_2	e_2	k_2	$\surd$
a_4	b_4	k_4	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	a_3	$\perp$	$\perp$	k_3	$\perp$	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	a_5	y_5	v_5	k_5	$\perp$	$\perp$	$\perp$	$\perp$	
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	a_6	y_6	e_6	k_6	

Risultato (II)

L1.A	L1.B	L1.K	L2.V	L2.K	L3.Y	L3.E	L3.K	
a_1	b_1	k_1	v_1	k_1	y_1	e_1	k_1	$\surd$
a_2	b_2	k_2	$\perp$	$\perp$	y_2	e_2	k_2	$\surd$

Si vede che la tabella del risultato (I) ha un numero maggiore di tuple che in realtà sono irrelevanti. Questo implica un tempo di risposta molto lunga rispetto alla seconda tabella. Inoltre, in termini di risorse, il calcolo del risultato (I) ha bisogno di *maggiore memoria* rispetto alla tabella del risultato (II). Si nota che la tabella (II) ha meno colonne che la tabella (I) perché gli schemi delle sue classi locali sono stati ridotti.

Il risultato finale della query, si ottiene facendo una proiezione:

$$\Pi_{\{A,Y,V\}}[\sigma_{\{B\ op\ c1 \wedge E\ op\ c2\}}(FJ_{\pounds})]$$

A	Y	V
a_1	y_1	v_1
a_2	y_2	$\perp$

e tale risultato verifica la correttezza del procedimento illustrato.

Capitolo 6

Conclusioni

Nella tesi è stato trattato il problema dell'ottimizzazione di query in MOMIS. In particolare, la tesi ha trattato l'ottimizzazione algebrica delle query contenenti operatori di outerjoin (full outerjoin, left/right outerjoin). Per risolvere questo problema sono stati utilizzati ed estesi i metodi di ottimizzazione sviluppati esplicitamente per gli operatore di outerjoin presentati in letteratura.

Le estensioni dei metodi presentati in letteratura hanno sfruttato le particolarità della query di full outerjoin che definisce una classe globale e le ipotesi fatte nel sistema MOMIS. Nella tesi abbiamo illustrato come queste particolarità ed ipotesi, diano luogo a molteplici possibilità di semplificazione, e quindi di ottimizzazione, delle query. Spesso queste ottimizzazioni sono molto significative, in quanto consentono sia di ridurre il numero di classi coinvolte nell'interrogazione sia di sostituire il full outerjoin con operazioni *più leggere*, quali il join.

Come lavoro futuro, si può delineare un approfondimento dei risultati ottenuti, allo scopo di ottenere nuove proprietà per le ottimizzazioni delle query di outerjoin in MOMIS. In particolare, queste proprietà di semplificazione possono derivare da una opportuna combinazione delle proprietà di semplificazione e di riduzione degli schemi, possibilità questa che nella tesi è stata accennata nell'ultimo capitolo. Molto probabilmente, uno studio più approfondito potrà portare anche ad una *migliore presentazione* dei risultati ottenuti nella tesi, ovvero ad una loro formulazione in forma più sintetica ed efficace.

Bibliografia

[1] CHEN, A.L.P. 1990 Outerjoin optimization in multidatabase systems. In 2nd International Symposium on databases in Parallel and Distributed systems.

[2] CODD, E.F 1979 Extending the relational database model to capture more meaning. ACM Trans. Database Syst.

[3] DATE, C.J 1986. An introduction on Database Systems, Vol. II. Addisons-Wesley, Reading, MA.

[4] GALINDO-LEGARIA, C.A. 1987. Rule-based optimization of database queries. Master's thesis, CINVESTAV-IPN, Mexico City (Sept). In Spanish.

[5] GALINDO-LEGARIA, C.A. 1992. Algebraic optimization of outerjoin queries. PhD thesis, Technical Rep. TR-12-92, Havard Univ., Cambridge, MA, May.

[6] GALINDO-LEGARIA, C.A. 1994. Outerjoins as disjunction. In Proceedings of ACM SIGMOD 1994 International Conference on Management of Data, (Minneapolis, MN), 348-358.

[7] GALINDO-LEGARIA, C.A. AND ROSENTHAL, A. 1992. How to extend a conventional optimizer handle-one and two side outerjoin.

[8] LACROIX,M. AND PIROTTE, A. 1976. Generalized joins. ACM SIGMOND.

[9] DAYAL, U. AND HWANG, H. 1984. View definition and generalization for database integration in a multidatabse system. IEEE Trans. Software.Eng.10,6(Nov),628-645.

[10] Rosenthal, A. And Galingo-Legaria C.A 1990. Query graph, implementing trees, and freely-reorderable outerjoins. In Proceding of ACM SIGMOND 1990 International Conference on Management of Data, (Atlantic City, NJ, May), 291-299.

[11] Wiederhold G. (1992). Mediators in the architecture of future information systems, Computer,Vol.25(3).p.38-49

[12] Subrahmanian V.S et al.(1995). HERMES: A hetereogeneous reasoning and mediator system. Technical report. Univ of Maryland.

[13] Bidauld A. et al.(2000). Repairing queries in a mediator approach. In 14^{th} European Conference on Artificial Intelligence.p.406-410.Berlin.

[14] Chawathe S.,Garcia-Molina et al.(1994). The TSIMMIS project: Integration of hetereogeneous information source. In proceedeings of IPSI conference, Tokyo Japan.

[15] Kerschberg L.(1997). The role of intelligent Agents in advanced Information System. In Advances in Databases. Vol. 1271, LNCS. C.

[16] Kimball R. & Mers R.(2000). the data WebHouse Toolkit: Building the Web-Enabled Data Warehouse. John Wiley & Sons, Inc.

[17] Friedman et al.(1997). Efficiently executing information-gathering plans. In 15^{th} International Conference on Artificial Intelligence.p.785-791, Nagoya. Japon.

[18] Kirk T. et al.(1995). The information Manifold. In proceeding of AAAI 1995 Spring Symposium on Information Gathering from Heteregeneous, Distribuited Environments,P.85-91.

[19] Mena E. et al.(1996). OBSERVER: An approach for query processing in global information.

[20] Silvia Zanni, Il componente Query Manager di MOMIS. Esecuzione di interrogazioni. Tesi di Laurea, 1999-2000, UNiversità di Modena e Reggio Emilia.

[21] A. Rabitti, Architettura di un mediatore per un sistema di sorgenti distribuite e autonome, 1997-1998, Università di Modena e Reggio Emilia.

[22] A. Zaccaria, MOMIS: il componente Query Manager. tesi di Laurea, 1997-1998, UNiversità di Modena.

[23] A. Zaccaria, MOMIS: il componente Query Manager. Tesi di Laurea, 1997-1998, Università di Modena e Reggio Emilia.

[24] M. Franceschini, Il Componente Query Manager di MOMIS: utilizzo della conoscenza estensionale. Tesi di Laurea 1999-2000, Università di Modena.

[25] S. castano and V. De Antonellis, Deriving Global Conceptual Views from Multiple Information Sources. In pre Proc of ER'97 Preconference Symposium an conceptual Modeling Historical Perspectives and Future Directions 1997.

[26] M. vincini, ODB-Qoptimizer, un'ottimizzazione semantico di interrogazioni. Tesi di Laurea 1994, Università di Modena.

[27] Anand Rajaraman and Jeffrey D.Ullman. Integration Information by outerjoins

and full disjunction ACME, 1997.

[28] K.E. Kostenarov, Elaborazione di interrogazioni in un sistema multi-database 2001-2002.

[29] Ronal Fagin. degrees of acyclicity for hypergraph and relational database schemes. J. ACM, 30:3, pp514:550 1983.