# Documentazione Aggiuntiva

Dopo aver prodotto lo script per la realizzazione della Base Dati (percorso: /SQL/Crea Database.sql), e dopo aver espresso la maggior parte dei vincoli tramite una condizione di check nel codice DCL, si riportano i vincoli (complessi) che sarà necessario codificare mediante l’uso di stored procedures e triggers (è possibile trovarli nel file /SQL/Triggers.sql).

Tali vincoli sono:

* Controllo della cardinalità minima pari ad uno per le associazioni “Afferenza Personale” e “Dirigenza”, lato Scuderia;
* Controllo della cardinalità minima pari ad uno per l’associazione “Afferenza Personale”, lato Personale;
* Controllo della professione per il personale che afferisce all’associazione “Dirigenza” (deve avere “Dirigente” come valore dell’attributo professione);
* Controllo sul corretto inserimento dei risultati, in particolare:
  + Devono essere inseriti solo blocchi di 20 risultati;
  + Devono essere inseriti risultati coerenti (stesso campionato e stessa pista);
  + I punteggi devono essere distinti, fatta eccezione per 0.
* Corretta eliminazione dei risultati, in particolare deve essere possibile cancellare solo i risultati dell’ultima gara inserita, e tale operazione comporta una cancellazione di tutti i risultati per quella gara. Tale operazione non deve però impedire lo svuotamento del database;
* Controllare che l’amministratore delegato di una scuderia faccia parte del personale di quella scuderia (per quel campionato).

Note:

* Per ogni inserimento nella tabella Risultati Attuali, è necessario verificare se ci sono nuovi record per quella pista e, in caso affermativo, allineare le informazioni nell’entità Piste;
* Si implementa, inoltre, un trigger che, terminato un campionato, provvede al corretto spostamento dei Risultati Attuali in Risultati Passati, e all’aggiornamento delle ridondanze per Scuderie e Piloti.

## Viste

Due delle operazioni eseguite con maggiore frequenza sono la visualizzazione delle classifiche piloti e la visualizzazione delle classifiche costruttori. Poiché tali informazioni non sono direttamente espresse dal database, ma ottenibili attraverso un insieme di query più o meno complesse, si è deciso di utilizzare lo strumento delle Viste per poter rappresentare questo dato. In particolare, tali viste permettono:

* La visualizzazione delle classifiche opportune senza la necessità di inviare al DBMS una query precisa. In tal modo si garantisce sia l’indipendenza dei dati visualizzati dal modo con cui essi vengono elaborati, sia la privatezza della struttura interna del DB (si fa riferimento ad un caso di utilizzo mediante software separato dal DBMS).  
  In questo senso, le viste rappresentano effettivamente lo schema esterno del sistema implementato.

Nota:

* Implementare le viste attraverso il costrutto MATERIALIZED VIEW permette di evitare il continuo ricalcolo delle query e di memorizzare il risultato nella forma di una vista materializzata, quindi salvata a tutti gli effetti in memoria. Questo velocizza le operazioni tenendo conto della frequenza con cui vengono svolte e dei dati che esse producono (pochi KB, è un costo sicuramente sostenibile). Sarà necessario inviare direttamente dei comandi di aggiornamento delle viste (REFRESH MATERIALIZED VIEW) negli opportuni trigger, con cadenza di 1 V/S e 1 V/A;
* L’implementazione attraverso il costrutto MATERIALIZED VIEW e il trigger di refresh ad esse associato hanno portato, in fase di testing, risultati inconsistenti sia con la logica del Database, sia con le modalità di funzionamento normalmente previste. Essendo tale costrutto una specifica di PostgreSQL e non dello standard SQL, per cui sarebbe necessario indagare l’effettiva realizzazione di una vista materializzata e il particolare modo di gestire i trigger in tale DBMS, si è deciso di lasciare le viste come semplici VIEW (ovvero virtuali, ricalcolate ad ogni richiesta), **ma** tenendo presente che, in virtù di una buona progettazione, è sicuramente preferibile avere una vista memorizzata (per i motivi di cui sopra).