# Case Study C North Island Racing Part Two

Title Page

Report Title : Case Study C – North Island Racing

Student Name : Anna Henson

Student Number : 1434929

Date Completed : 29/10/2017

Total Number Of Hours Taken : 59 hours

Table of Contents

[Introduction to the Report 3](#_Toc496971256)

[4.1 Data volume map 4](#_Toc496971257)

[4.2 Data usage 5](#_Toc496971258)

[4.3 Logical level business function to physical design technique matrix 7](#_Toc496971259)

[4.4 Physical Design Techniques that can be used for each Process as defined by the Business Function to Physical Design Technique Matrix 9](#_Toc496971260)

[4.5 Final ERD and standard relation notation to reflect any changes to entities and/or the attributes 18](#_Toc496971261)

[4.6 SQL Queries needed to implement data retrieval operational business rules for each process 20](#_Toc496971262)

[4.7 Rewritten data usage maps 25](#_Toc496971263)

[4.8 Data Dictionary 27](#_Toc496971264)

# Introduction to the Report

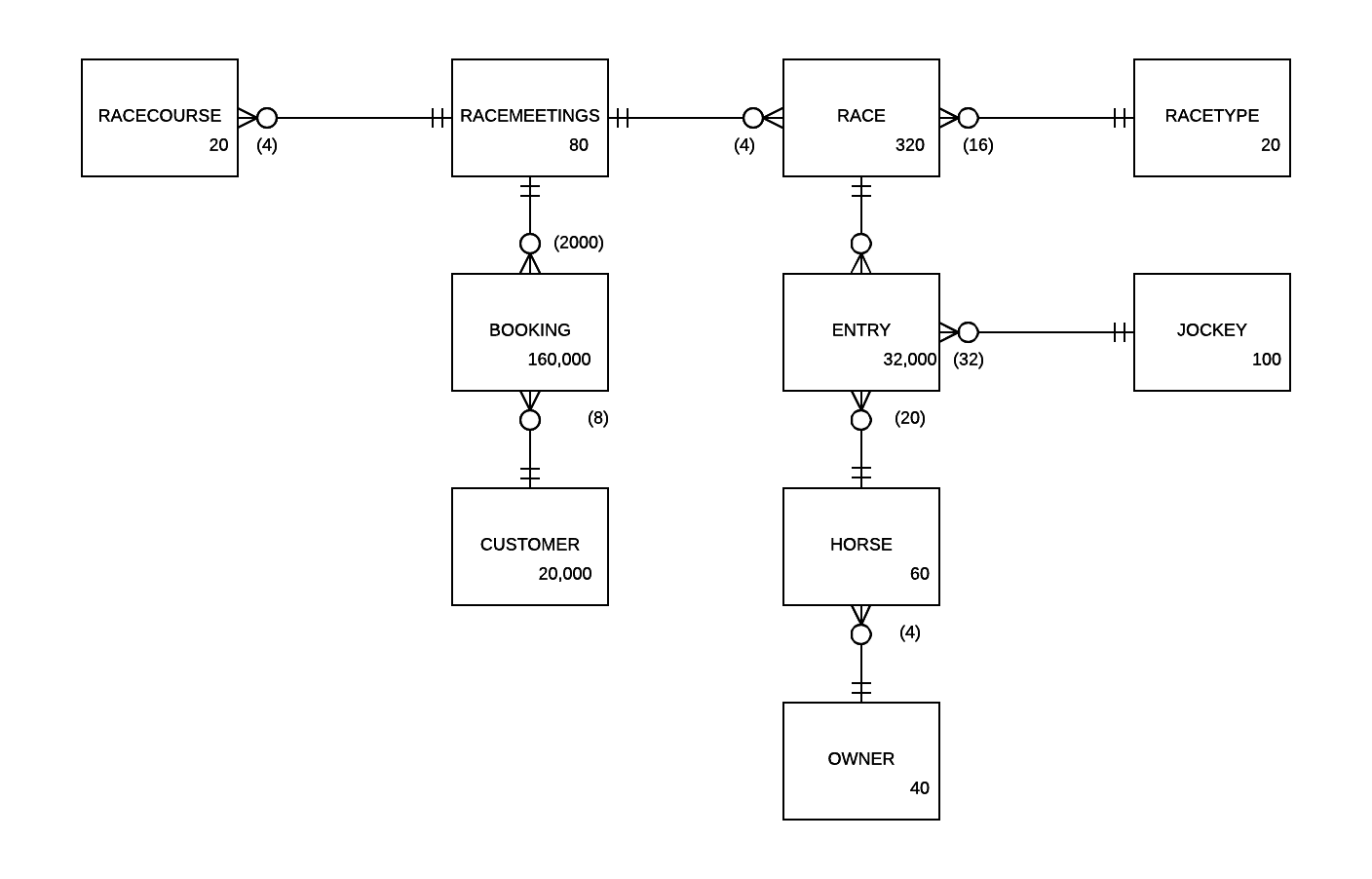
This report is based on the physical design requirements which is needed to translate the logical description of data into a design that will provide adequate performance. This former process was driven and directed by the goals alongside the requirements of the business, and ensures that the database will meet performance, integrity, security and recoverability specifications as required by North Island Racing for administration purposes.

We are taking information from the logical model which evolved around the needs of North Island Racing, and its business processes along with the categories of data. This allows progression to the physical design as to convert the logical modal, which reflected the inherent nature of the business.

The physical database design is the one which is intended to be implemented in a database management system. It will include detailed specification such as indexing options, partitioning and other parameters which will reside in the DBMS data dictionary. So, the data which the business wishes to maintain, or retrieve will be reflected within the technical specifications derived from the logical design covered in part one of the previous report.

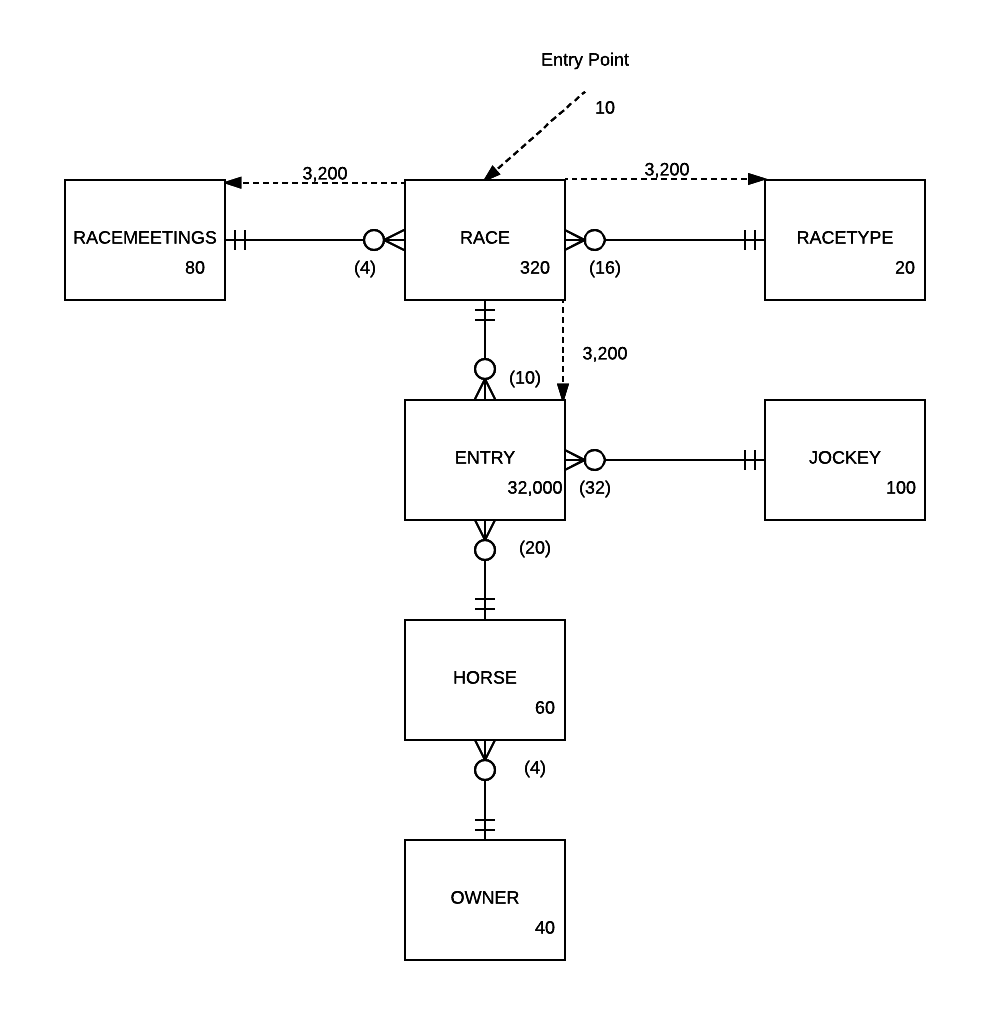
In conclusion the physical design will specify the physical configuration as to ensure that all the pieces come together and allow a process to shape and define a database which is driven by business’s specification’s. To make this information clearer section two examines each process under a subsection with a heading, which will examine any appropriate physical design techniques to be used for that process.

4.1 Data volume map



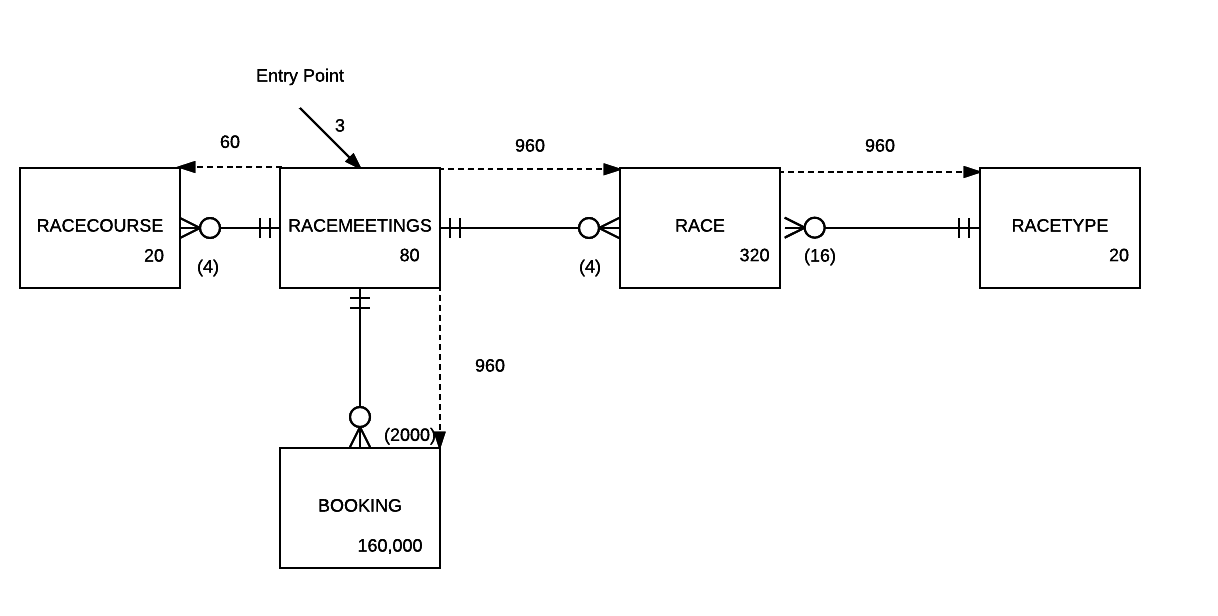
### 4.2 Data usage maps

#### Race report



(10)

#### Meeting report



### 4.3 Logical level business function to physical design technique matrix

|  | One to one relationship | One to many with reference data | Associate entity with non-key attributes | Duplication on non-key attributes | Indexing | Vertical Partitioning | Horizontal Partitioning |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Enter, Modify or Delete Horses | N | N | N | N | Y | N | N |
| Enter, Modify or  Delete Customers | N | N | N | N | Y | N | N |
| Enter, Modify or Delete Meetings | N | N | N | N | Y | N | N |
| Enter, Modify of Delete Bookings | N | N | N | Y | Y | N | N |
| Enter a race and register a horse on to  a race | N | Y | N | N | Y | N | N |
| Record a horse’s  time for a race | N | N | N | Y | Y | N | N |
| Races Report | N | Y | N | N | Y | N | N |
| Meetings Report | N | Y | N | Y | Y | N | N |

Denormalization techniques that cannot be used.

* We cannot use the one to one denormalization technique because we do not have any one to one relationships in this design.
* Vertical partitioning cannot be used because we do not a set of columns that are used in one set of processes.
* Associate entity with non-key attributes also cannot be used as it requires a bridge table with non-key attributes and this design has no bridge tables.
* Horizonal partitioning has been considered for process seven, but this would make process eight more complicated so has been dismissed.

### 4.4 Physical Design Techniques that can be used for each Process as defined by the Business Function to Physical Design Technique Matrix

#### Process One Enter, Modify or Delete a Horse.

The one physical design technique which can be used for this process according to the matrix is indexing. We can index this process as the whole point of indexing is to speed up the searches or queries and can achieve this by cutting down the number of rows which the query will need to search in the horse table. We won’t index every attribute but create the index on a column from one table, and indexing will determine the location of rows as to satisfy some condition for specific values as requested by searches or queries. This involves the columns horse name plus the owners first, and last name as they are the columns needed and stipulated to be ordered by the horse name and owners last name in the first SQL query. Both queries use order by and one has a join to two tables. We will need indexes seven and eight.

There is no option to use a one-to-one physical denormalization technique for this or other processes as we do not have this relationship in this design. Another option we can’t use for example is the associate entity with non-key attributes as the design requires a bridge table with non-key-attributes, and this design has no bridge tables for this or any of the other processes. Vertical partitioning will not be an option because we do not a set of columns that are used in one set of processes and horizontal is not an option and there is no defined subset of rows used in this process.

In conclusion we will only index those attributes which are necessary for the queries needed to implement the data retrieval concerning operational business rules for this process. After careful consideration it is decided that indexing will be used for this process as it will help to optimise data retrieval to speed up searches or queries regarding maintenance. Even though maintenance performance may a bit slower, but far more importance has been put on processes seven and eight which is two reports.

The indexes I have added are index 7 which is the horse name and the owner id on the horse table. The owner id is used in the join to the owner table and horse name is used in the order by. Index 8 has been added because we order the owners by owner last name and first name.

#### Process 2 Enter, Modify or Delete Customers

There is no option to use a one-to-one physical denormalization technique for this or other processes as we do not have this relationship in this design. Another option we can’t use for example is the associate entity with non-key attributes as the design requires a bridge table with non-key-attributes, and this design has no bridge tables for this or any of the other processes. Vertical partitioning will not be an option because we do not a set of columns that are used in one set of processes and horizontal is not an option and there is no defined subset of rows used in this process.

The physical design technique which according to the physical design technique matrix is indexing. Once again, we want to speed up any searches or queries, and the advantage is that it will be quicker to search the index than the whole table. We can cut down the number of rows to search on the customer table for the specific values needed to satisfy some condition through the SQL queries. We take advantage of not indexing every attribute but create an index on a column in one table and use those column values stored within the North Island Racing DBMS. Again, the second process needs specific columns which are Customer identification, last and first name, street address, suburb, city, phone number, email address, and their credit status. A second query wants a booking identification, booking date and the quantity. There is a where clause in the first query, and both use the order by to sort the data. To for fill the needs of this process we will use indexes four and three.

Disadvantage is to make sure that we don’t index every attribute which will slow down any searches or queries and reduce the efficient location of rows needed to optimise this process. Secondly it takes more disk space.

After careful consideration it is decided that indexing will be used as maintenance of processes is not as important as that of process seven and eight. The latter have been highlighted as the more important by North Island Racing for two reports one being finished races report, and the other a meetings report which are run frequently.

The indexes I have added are index three, used to order the booking rows by date with in a customer. Index four is used for the ordering of the customer data by last name and then first name.

#### Process 3 Enter, Modify or Delete Meetings

There is no option to use a one-to-one physical denormalization technique for this or other processes as we do not have this relationship in this design. Another option we can’t use for example is the associate entity with non-key attributes as the design requires a bridge table with non-key-attributes, and this design has no bridge tables for this or any of the other processes. Vertical partitioning will not be an option because we do not a set of columns that are used in one set of processes and horizontal is not an option and there is no defined subset of rows used in this process.

The physical design technique which according to the physical design technique matrix is indexing. Once again, we want to speed up any searches or queries, and the advantage is that it will be quicker to search the index than the whole relation. We can cut down the number of rows to search on the race meeting table for the specific values needed to satisfy some condition through the SQL queries. We take advantage of not indexing every attribute but create an index on a column in one table and use those column values stored within the North Island Racing DBMS.

The advantage is we cut down the number of rows rather than searching every single row in the table and will reduce the amount of data to search or examine within the table. This helps the performance of storing and connecting files, which then improves and optimises the databases ability to search for the appropriate data as required by the administrator. The columns required for the first query are the meeting identification, meeting name, race course name, meeting capacity, meeting status, and the meeting date. The second query columns are race course name, street address, suburb, city, and phone number. Other field needed are race identifier, race name, booking identifier, customer identifier, booking date, and quantity. To achieve this goal indexes 2, 3, and 5 will be used.

Disadvantage is to make sure we don’t index every attribute which will slow down any searches or queries and reduce the efficient location of rows needed to optimise this process. Another disadvantage is that more disk space will be used to store the index.

After careful consideration it has been decided that we will use indexing. It will improve the retrieval of appropriate data for implementing processes seven and eight. After careful consideration we will use indexing as it easier to search an index rather than the whole relation within or across the table.

#### Process 4 Enter, Modify or Delete Bookings

There is no option to use a one-to-one physical denormalization technique for this or other processes as we do not have this relationship in this design. Another option we can’t use for example is the associate entity with non-key attributes as the design requires a bridge table with non-key-attributes, and this design has no bridge tables for this or any of the other processes. Vertical partitioning will not be an option because we do not a set of columns that are used in one set of processes and horizontal is not an option and there is no defined subset of rows used in this process.

The physical design technique which can be used according to the physical design technique matrix is indexing. Once again, we want to speed up any searches or queries, and the advantage is that it will be quicker to search the index than the whole relation. We can cut down the number of rows to search on the booking table for the specific values needed to satisfy some condition through the SQL queries. We take advantage of not indexing every attribute but create an index on a column in one table and use those column values stored within the North Island Racing DBMS. For process four we need the following columns customer identifier, customer last and first name, street address, suburb, city, phone number, email address, and their credit status. Others needed are the meeting identifier, meeting name, race course name, meeting status, capacity and the meeting date. The other columns are booking identifier, meeting identifier, booking date, quantity. Both where and order by clauses are used in the four queries. We will use indexes two, three and six.

The advantage is we cut down the number of rows rather than searching every single row in the table and use those stored as it will reduce the amount of data to search or examine across the table.

Disadvantage is if we index every attribute it will slow down any searches or queries and reduce the efficient location of rows needed to optimise this process. Secondly another disadvantage more disk space is used to store the index. However, the overriding factor is to improve the performance of processes seven and eight rather than the those which are considered maintenance.

After careful consideration it has been decided that we will use indexing. It will improve the retrieval of appropriate data for the most important two processes seven and eight. After careful consideration we will use indexing for this process four to optimise searches and queries.

#### Process 5 Enter a race and register a horse onto a race

There is no option to use a one-to-one physical denormalization technique for this or other processes as we do not have this relationship in this design. Another option we can’t use for example is the associate entity with non-key attributes as the design requires a bridge table with non-key-attributes, and this design has no bridge tables for this or any of the other processes. Vertical partitioning will not be an option because we do not a set of columns that are used in one set of processes and horizontal is not an option and there is no defined subset of rows used in this process.

The physical design technique which can be used according to the physical design technique matrix is indexing. Once again, we want to speed up any searches or queries, and the advantage is that it will be quicker to search the index than the whole relation. We can cut down the number of rows to search on both the horse and race table for the specific values needed to satisfy some condition through the SQL queries. We take advantage of creating an index on a column in one table and use those column values stored within the North Island Racing DBMS. Once again, we need specific columns as follows meeting identifier, meeting name, race course name, status, capacity, and meeting date. Others are race identifier, race name, race time, status, race type description, and meeting name. Next is the horse identifier, horse name, gender, date of birth, owners last name and first name, race type identifier plus the description. Then jockey ID, their first and last name, weight along with date of birth, entry id, horse name, last name, first name, race name, meeting name and the meeting date. We will indexes two, five, seven and eight.

The advantage is we cut down the number of rows rather than searching every single row in the table as to reduce the amount of data a query examines within the table. Another advantage will be to access individual rows as they will have a specific value stored in the index.

An opportunity does seem possible to use a physical design technique called de-normalisation of one-to-may reference data, which means we take the race type description and move it into the race table. This will benefit process seven and eight as well as this process as they no longer need to join to the race type table to get the description. The race type table would be removed from the design as it is no longer needed.

If we look at the disadvantages, then we could lose data integrity so must ensure that if we need to update a race description we now need to update every row in the race table that has the old value, where before we would just update it once in the race type table.

The decision is to use de-normalisation of one-to-may reference data as a way of improving performance through not having to access both the race and race type tables and just use the race type description from the race table.

#### Process 6 Record a horse’s time for a race

There is no option to use a one-to-one physical denormalization technique for this or other processes as we do not have this relationship in this design. Another option we can’t use for example is the associate entity with non-key attributes as the design requires a bridge table with non-key-attributes, and this design has no bridge tables for this or any of the other processes. Vertical partitioning will not be an option because we do not a set of columns that are used in one set of processes and horizontal is not an option and there is no defined subset of rows used in this process.

The physical design technique which can be used according to the physical design technique matrix is indexing. Once again, we want to speed up any searches or queries, and the advantage is that it will be quicker to search the index than the whole table. We can cut down the number of rows to search on both the horse and race table for the specific values needed to satisfy some condition through the SQL queries. We take advantage of creating an index on a column or columns in one table and use those column values stored within the North Island Racing DBMS.

We need to make sure we don’t index every attribute which will slow down any inserts or updates and reduce the efficient location of rows needed to optimise this process. Secondly another disadvantage is more disk space will be used for storing the index.

After careful consideration we will use indexing for searches and queries, which helps maintenance as is the case of all the processes except for seven and eight. The latter two are emphasised as the two most important processes for reading the data as opposed to maintenance of data.

An opportunity does seem possible to use a physical design technique called duplication, which means we take advantage of copying across a non-key attribute across one join only in another table. We can then take advantage that this data relationship to allow maintenance of data in a single field, whilst still only having to locate data on one row within the table as not to take up more disk space for storage. In this case we will copy the jockey name to the entry table to benefit process seven. It will make this process slightly more complex as it must copy the jockey name to the entry table, but since process seven is considered important I think this acceptable.

If we look at the disadvantages, then we could lose data integrity. If we need to update the jockey name, we now have to update the jockey table and every row in the entry table that has the old jockey’s name.

#### Process 7 Races Report

There is no option to use a one-to-one physical denormalization technique for this or other processes as we do not have this relationship in this design. Another option we can’t use for example is the associate entity with non-key attributes as the design requires a bridge table with non-key-attributes, and this design has no bridge tables for this or any of the other processes. Vertical partitioning will not be an option because we do not a set of columns that are used in one set of processes.

Looking at the matrix we can see that under the physical design considerations we have an opportunity to use one-to-many reference data technique to improve the performance for process 7. This relationship is between race and race type of which the latter table is considered reference data.

We have two queries to consider for these reports. They need the following fields raceid, race name, time, status, race type description, meeting name, date, horse name, horse time, jockey name, and owner name.

RACE(RaceID, RaceName, StreetAddresss, Suburb, City, PhoneNumber)

RACETYPE(RaceTypeID, Description)

At the table level we consider race type to be reference data and has only a relationship with one other table called race as it exists on the one-side of a one-to-many relationship. Processes five, seven, and eight use description from the race type table, and race type is only used as the ‘one’ in a one-to-many relationship for this database design, which allows it to be de-normalised into the race table. There is no process that updates the data in the race type table.

By doing this technique we are denormalising the race type entity into the race entity, and now gives us the following new relation for the entity race:

RACE(RaceID, RaceName, Time, Status, RaceTypeDescription, RaceMeetingID\*)

Looking at the data usage map for the races report the daily average number of accesses for the join is 10 giving a total of 32,000 accesses. The performance of the report would be enhanced because the join to the Race Type table is removed. The case study c information says that race type data is unlikely for change in the future. Therefore, the likely hood of the data getting out of date is low, but if they did ever need to change a race type description they would have to update all the rows in the race table with the old value. Because this report is one of the two most important processes and after careful consideration it has been decided that the race type entity is to be de-normalised into the race entity.

An opportunity does seem possible to use a physical design technique called duplication, which means we take advantage of copying across a non-key attribute across one join in another table. Since this can only be done across one join the only candidate seems to be duplicating the Jockey name in the entry table. We could also do the horse name, but since we need the horse id to be able to join to the owner table it doesn’t seem sensible. Looking at the data usage map this would remove 32,000 access of the jokey table each time the report is run.

The disadvantage with this technique is that if the jockey name needs updating all the rows in the entry table with the old jockey name need to be updated. It also will take slightly more disk space. JockeyID needs to be left in the entry table to allow enforcing the ON DELETE RESTRICT constraint.

Duplication will make creating and updating a entry more complex, but since this report is considered a important process this is deemed acceptable.

If we use horizontal partitioning then we will be splitting a table into two tables that will contain a subset of rows that were in the initial table, which for this process is the race table. This means that there would be race scheduled and race finished tables and split the race table into two segments, and usually query only the rows of one partition. Advantage is that horizontal partitioning can be applied when querying only one partition for example the active rows, and any full scans of the data especially if there are only a few active rows will be faster. In essence we get physical optimisation rather than optimisation at the conceptual level.

The disadvantage with doing this is it will make process eight more complete as we would need to union both table partitions to satisfy the report requirements. Since the process eight is also considered an import process I have decided not to use horizontal partitioning.

Indexing can be used to help with this process. I’ve created index nine which indexes the status field as it is used as the filter in the where clause, it also has MeetingID which is used in the join. Index seven will also be used to join the horse table to the owner table. The last index for this process is 10 which is used to join to the horse table.

#### Process 8 Meetings Report

There is no option to use a one-to-one physical denormalization technique for this or other processes as we do not have this relationship in this design. Another option we can’t use for example is the associate entity with non-key attributes as the design requires a bridge table with non-key-attributes, and this design has no bridge tables for this or any of the other processes. Vertical partitioning will not be an option because we do not a set of columns that are used in one set of processes.

According to the matrix it indicates an option to utilise indexing. Once again, we want to speed up any searches or queries, and the advantage is that it will be quicker to search the index than the whole relation. We can cut down the number of rows to search to retrieve data for the specific values needed to satisfy some condition through the SQL queries. We take advantage of not indexing every attribute but create an index on a column in one table and use those column values stored within the North Island Racing DBMS**.**

The advantage is we cut down the number of rows rather than searching every single row in the table and use those stored as it will reduce the amount of data to search or examine within the table. This helps the performance of storing and connecting files, which then improves and optimises the databases ability to search for the appropriate data as required by the administrator. Another advantage will be to access individual attributes as they will have a specific value stored on the database and improve the reports.

Disadvantage is to make sure we don’t index every attribute which will slow down any searches or queries and reduce the efficient location of rows needed to optimise this process. Secondly disadvantage is that indexing every attribute can use unnecessary disk space rather than it being optimised for storage. The trade-off is slower performance for maintenance of data for inserting, deleting or updating of records.

After careful consideration processes seven and eight to read data for two reports is more important. It has been emphasised that process seven finished races report and the meetings report must take priority over the maintenance processes for the physical design. The primary function of this database is to improve the retrieval of data.

They indexes I have added are index five which has the meeting id indexed on the race table. This is used to join to the race table. Index two will be used to join to the race course table. Lastly index eleven is the MeetingID of the booking table for use in the count subquery.

We can also see from the logical level business function process to physical design technique matrix a chance to use one-to-many reference data, which will be on a non-key attribute called description in the race type table.

If we look at the disadvantages, then we could lose data integrity so must ensure that if we need to update the race type description ever row with the old description is updated.

The decision is to use duplication as a way of improving performance through not having to access both the race and race type tables, and just find the non-key attribute race type description on the race table.

Horizonal partitioning for process seven was dismissed as it would make this process more complex requiring a union between the two partitions to satisfy this process.

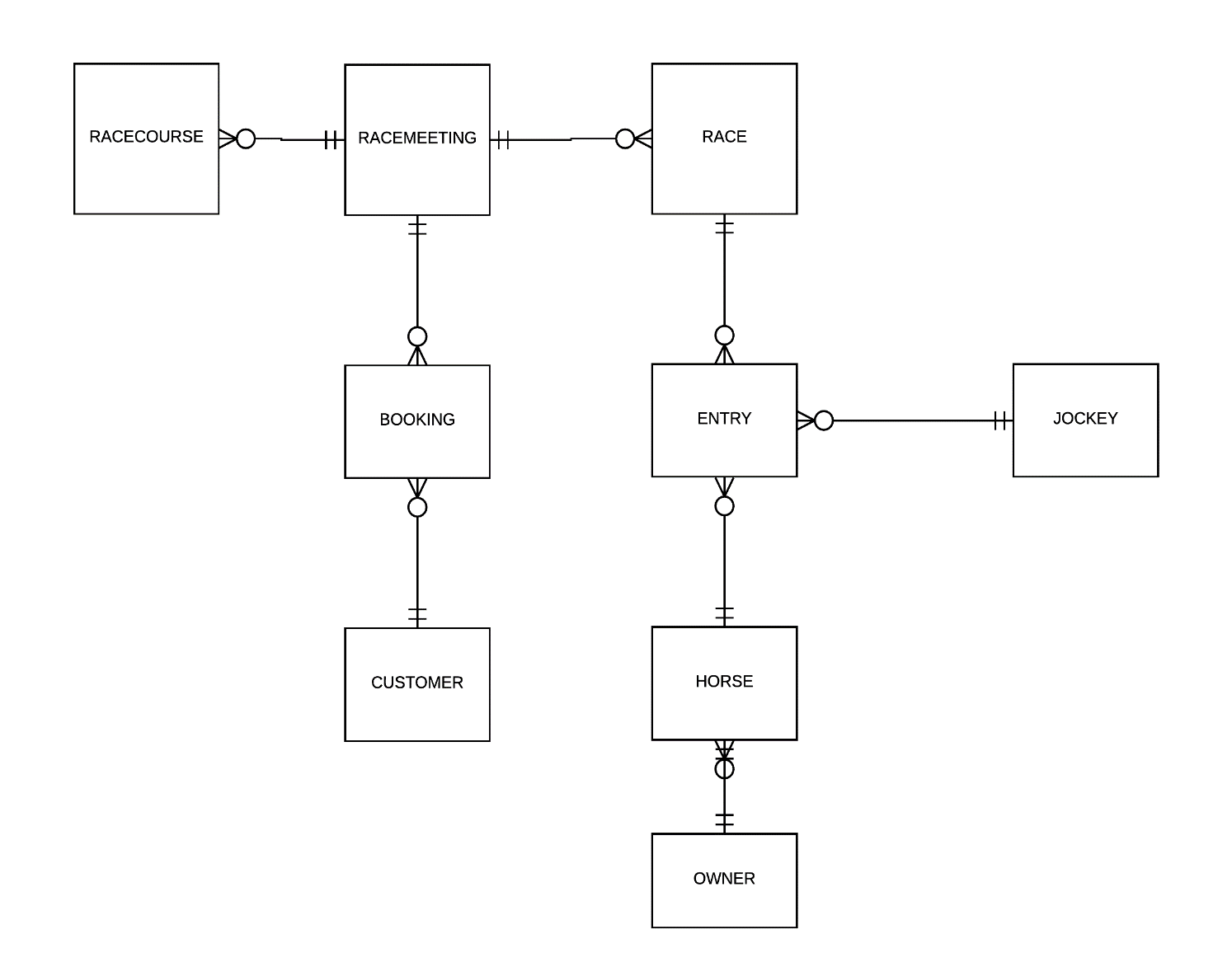
The query now uses four tables

RACE(RaceID, RaceName, Time, Status, RaceTypeDescription, RaceMeetingID\*)

RACEMEETING(RaceMeetingID, MeetingName, Capacity, Status, MeetingDate, RaceCourseID\*)  
RACECOURSE**(**RaceCourseID, RaceCourseName, SrteetAddress, Suburb, City, PhoneNumber )  
BOOKING(BookingID, BookingDate, Quantity, RaceMeetingID\*, CustomerID\*)

Since RaceTypeDescription has been de-normalised into the race table we will save ourselves 960 joins

### 4.5 Final ERD and standard relation notation to reflect any changes to entities and/or the attributes



CUSTOMER **(**CustomerID**,** LastName, FirstName, StreetAddress, Suburb, City, PhoneNumber, EmailAddress, CreditStatus)

BOOKING(BookingID, BookingDate, Quantity, RaceMeetingID\*, CustomerID\*)

CustomerID: ON DELETE CASCADE

RaceMeetingID: ON DELETE RESTRICT

RACECOURSE**(**RaceCourseID, RaceCourseName, SrteetAddress, Suburb, City, PhoneNumber )

RACEMEETING(RaceMeetingID, MeetingName, Capacity, Status, MeetingDate, RaceCourseID\*)

RaceCourseID: ON DELETE RESTRICT

RACE (RaceID, RaceName, Time, Status, RaceTypeDescription, RaceMeetingID\*)

RaceMeetingID: ON DELETE RESTRICT

ENTRY(EntryID, RaceID\*, HorseID\*, JockeyID \*, JockeyFirstName, JockyLastName, Status, HorseTime )

RaceID: ON DELETE RESTRICT

JockeyID: ON DELETE RESTRICT

HorseID: ON DELETE RESTRICT

HORSE(HorseID, HorseName, Gender, DateOfBirth, OwnerID\*)

OwnerID : ON DELETE RESTRICT

JOCKEY(JockeyID, LastName, FirstName, Weight, DateOfBirth, EmailAddress)

OWNER(OwnerID, LastName, FirstName, StreetAddress, Suburb, City, PhoneNumber, EmailAddress, Status)

### 4.6 SQL Queries needed to implement data retrieval operational business rules for each process

#### Enter modify or delete horses.

SELECT HORSEID, HORSENAME, GENDER, DATEOFBIRTH, LASTNAME, FIRSTNAME  
FROM HORSE, OWNER  
WHERE HORSE.OWNERID = OWNER.OWNERID  
ORDER BY HORSENAME, LASTNAME;

SELECT OWNERID, LASTNAME, FIRSTNAME, STREETADDRESS, SUBURB, CITY, PHONENUMBER, EMAILADDRESS  
FROM OWNER  
ORDER BY FIRSTNAME, LASTNAME;

#### Enter, modify or delete customers

SELECT CUSTOMERID, LASTNAME, FIRSTNAME, STREETADDRESS, SUBURB, CITY, PHONENUMBER, EMAILADDRESS, CREDITSTATUS  
FROM CUSTOMER  
ORDER BY LASTNAME, FIRSTNAME;

SELECT BOOKINGID, BOOKINGDATE, QUANTITY  
FROM BOOKING  
WHERE CUSTOMERID = 50001  
ORDER BY BOOKINGDATE;

#### Enter, modify or delete meetings

SELECT MEETINGID, MEETINGNAME, RACECOURSE.RACECOURSENAME, MEETING.CAPACITY, MEETING.STATUS, MEETING.MEETINGDATE  
FROM MEETING, RACECOURSE  
WHERE MEETING.RACECOURSEID = RACECOURSE.RACECOURSEID  
ORDER BY MEETINGNAME;

SELECT RACECOURSENAME, STREETADDRESS, SUBURB, CITY, PHONENUMBER  
FROM RACECOURSE  
ORDER BY RACECOURSENAME;

SELECT RACEID, RACENAME  
FROM RACE  
WHERE MEETINGID = 3001  
ORDER BY RACENAME;

SELECT BOOKINGID, CUSTOMERID, BOOKINGDATE, QUANTITY  
FROM BOOKING  
WHERE MEETINGID = 3001  
ORDER BY BOOKINGDATE;

#### Enter modify or delete bookings

SELECT CUSTOMERID, LASTNAME, FIRSTNAME, STREETADDRESS, SUBURB, CITY, PHONENUMBER, EMAILADDRESS, CREDITSTATUS  
FROM CUSTOMER  
ORDER BY LASTNAME, FIRSTNAME;

SELECT MEETINGID, MEETINGNAME, RACECOURSE.RACECOURSENAME, MEETING.STATUS, MEETING.CAPACITY, MEETING.MEETINGDATE  
FROM MEETING, RACECOURSE  
WHERE MEETING.RACECOURSEID = RACECOURSE.RACECOURSEID  
ORDER BY MEETINGNAME;

SELECT BOOKINGID, MEETINGID, BOOKINGDATE, QUANTITY  
FROM BOOKING  
WHERE CUSTOMERID = 50002  
ORDER BY BOOKINGDATE;

SELECT BOOKINGID, CUSTOMERID, BOOKINGDATE, QUANTITY  
FROM BOOKING  
WHERE MEETINGID = 3001  
ORDER BY BOOKINGDATE;

#### Enter a race and register a horse onto a race

SELECT MEETINGID, MEETINGNAME, RACECOURSE.RACECOURSENAME, MEETING.STATUS, MEETING.CAPACITY, MEETING.MEETINGDATE  
FROM MEETING, RACECOURSE  
WHERE MEETING.RACECOURSEID = RACECOURSE.RACECOURSEID  
ORDER BY MEETINGNAME;

SELECT RACEID, RACENAME, RACETIME, RACE.STATUS, RACETYPEDESCRIPTION, MEETING.MEETINGNAME  
FROM RACE, MEETING  
WHERE RACE.MEETINGID = MEETING.MEETINGID  
ORDER BY RACENAME;

SELECT HORSEID, HORSENAME, GENDER, DATEOFBIRTH, LASTNAME, FIRSTNAME  
FROM HORSE, OWNER  
WHERE HORSE.OWNERID = OWNER.OWNERID  
ORDER BY HORSENAME, LASTNAME;

SELECT JOCKEYID, LASTNAME, FIRSTNAME, WEIGHT, DATEOFBIRTH  
FROM JOCKEY  
ORDER BY LASTNAME, FIRSTNAME

SELECT ENTRYID, HORSENAME, JOCKEYLASTNAME AS JOCKEY\_LAST\_NAME, JOCKEYFIRSTNAME AS JOCKEY\_FIRST\_NAME, RACENAME, MEETINGNAME, MEETINGDATE  
FROM ENTRY, RACE, HORSE, MEETING, JOCKEY  
WHERE ENTRY.RACEID = RACE.RACEID  
 AND ENTRY.HORSEID = HORSE.HORSEID  
 AND RACE.MEETINGID = MEETING.MEETINGID  
 AND RACE.RACEID = 70001  
ORDER BY RACENAME;

#### Record a horse's time for a race

SELECT RACEID, RACENAME, RACETIME, RACE.STATUS, MEETING.MEETINGNAME  
FROM RACE, MEETING  
WHERE RACE.MEETINGID = MEETING.MEETINGID  
ORDER BY RACENAME;

SELECT ENTRYID, HORSENAME, JOCKEYLASTNAME AS JOCKEY\_LAST\_NAME, JOCKEYFIRSTNAME AS JOCKEY\_FIRST\_NAME, RACENAME, MEETINGNAME, MEETINGDATE  
FROM ENTRY, RACE, HORSE, MEETING  
WHERE ENTRY.RACEID = RACE.RACEID  
 AND ENTRY.HORSEID = HORSE.HORSEID  
 AND RACE.MEETINGID = MEETING.MEETINGID  
 AND RACE.RACEID = 70001  
ORDER BY RACENAME;

#### Scheduled races report

SELECT RACE.RACEID, RACENAME, RACETIME, RACETYPEDESCRIPTION AS RACE\_TYPE, MEETINGNAME, MEETINGDATE, HORSENAME, HORSETIME, JOCKEYLASTNAME AS JOCKEY\_LAST\_NAME, JOCKEYFIRSTNAME AS JOCKEY\_FIRST\_NAME, OWNER.LASTNAME AS OWNER\_LAST\_NAME, OWNER.FIRSTNAME AS OWNER\_FIRSTNAME

FROM RACE, ENTRY, HORSE, OWNER, MEETING

WHERE RACE.RACEID = ENTRY.RACEID

AND ENTRY.HORSEID = HORSE.HORSEID

AND HORSE.OWNERID = OWNER.OWNERID

AND RACE.MEETINGID = MEETING.MEETINGID

AND RACE.STATUS = 'Scheduled';

#### Finished races

SELECT RACE.RACEID, RACENAME, RACETIME, RACETYPEDESCRIPTION AS RACE\_TYPE, MEETINGNAME, MEETINGDATE, HORSENAME, HORSETIME, JOCKEYLASTNAME AS JOCKEY\_LAST\_NAME, JOCKEYFIRSTNAME AS JOCKEY\_FIRST\_NAME, OWNER.LASTNAME AS OWNER\_LAST\_NAME, OWNER.FIRSTNAME AS OWNER\_FIRSTNAME

FROM RACE, ENTRY, HORSE, OWNER, MEETING

WHERE RACE.RACEID = ENTRY.RACEID

AND ENTRY.HORSEID = HORSE.HORSEID

AND HORSE.OWNERID = OWNER.OWNERID

AND RACE.MEETINGID = MEETING.MEETINGID

AND RACE.STATUS = 'Finished';

#### MEETING REPORT

SELECT MEETING.MEETINGID, MEETINGNAME, MEETING.STATUS, CAPACITY, MEETINGDATE, RACECOURSENAME, STREETADDRESS, RACENAME, RACETYPEDESCRIPTION, (SELECT COUNT(\*) FROM BOOKING WHERE BOOKING.MEETINGID = MEETING.MEETINGID) AS NUMBER\_OF\_BOOKINGS

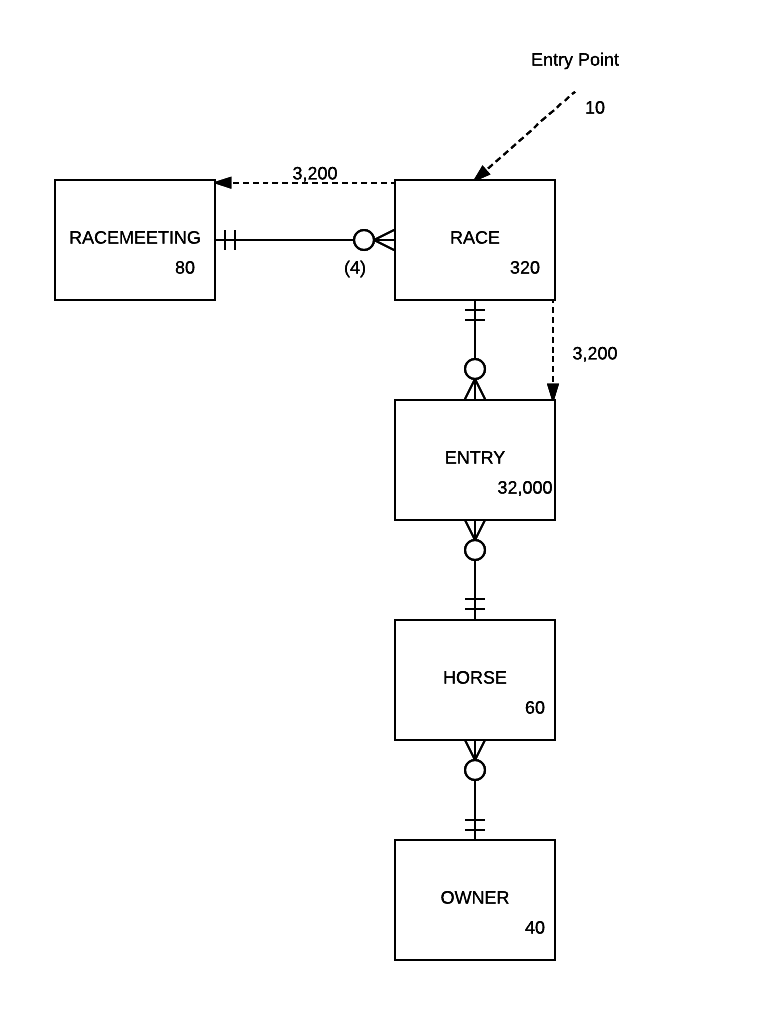
FROM MEETING, RACE, RACECOURSE

WHERE MEETING.MEETINGID = RACE.MEETINGID

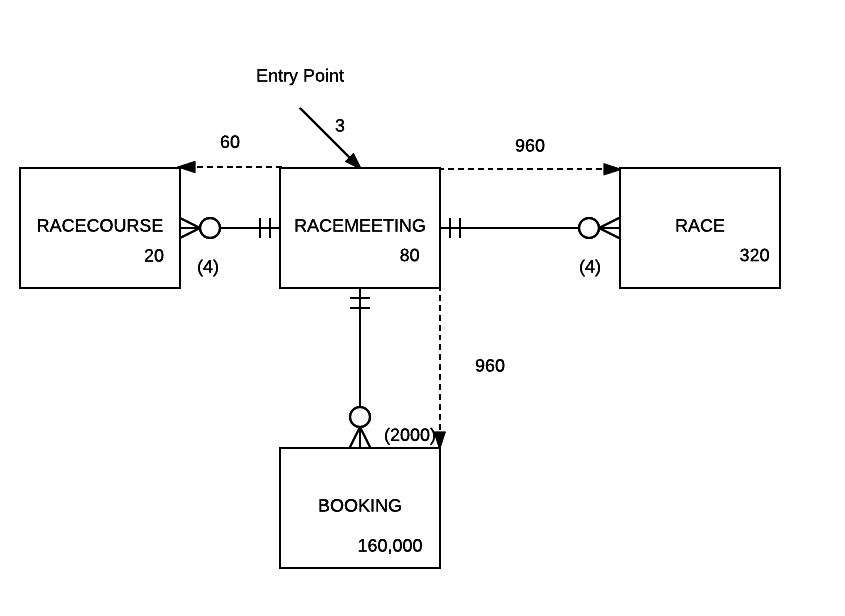
AND MEETING.RACECOURSEID = RACECOURSE.RACECOURSEID

### 4.7 Rewritten data usage maps

#### Races report



#### Meeting report



### 4.8 Data Dictionary

**RACECOURSE TABLE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data type & size** | **Domain(all ranges are inclusive)**  **, example or format** | **Primary**  **Key** | **Foreign**  **Key** | **Required** | **Default value** | **Index** |
| RaceCourseID | The identifier of the race course | NUMBER(5) | 1-99999 | Yes | - | Yes | Auto-number | Auto |
| RaceCourseName | The Name of the racecourse | VARCHAR2(40) | Example : ‘ Ascot park ‘ | - | - | Yes | - | Index 1 |
| StreetAddress | The Street address of the race course | VARCHAR2(40) | Example : ‘ 78 Ascot Road ‘ | - | - | Yes | - | - |
| Suburb | The Suburb of race the course | VARCHAR2(20) | Example : ‘ Ascot ‘ | - | - | Yes | - | - |
| City | The City of the racecourse | VARCHAR2(30) | Example : ‘ Masterton ‘ | - | - | Yes | - | - |
| PhoneNumber | The Phone number of the race course | VARCHAR(12) | Example : ‘ 7891237 ‘ | - | - | Yes | - | - |

**MEETING TABLE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data type & size** | **Domain(all ranges are inclusive)**  **, example or format** | **primary key** | **Foreign key** | **Required** | **Default value** | **Index** |
| MeetingID | The Identifier of the race meeting | NUMBER(5) | 1-99999 | Yes | - | Yes | Auto-number | Auto |
| MeetingName | The Name of race meeting | VARCHAR2(40) | Example : ‘ Ascot challenge ‘ | - | - | Yes | - | Index 2 |
| Capacity | The capacity of the race meeting | Number(6) | Example : ‘ 207 ‘ | - | - | Yes | - | - |
| Status | The Status of the race meeting | VARCHAR2(11) | C for ‘ Confirmed’ or U for ‘Unconfirmed’ or CA for ‘ Cancelled’ | - | - | Yes | - | - |
| MeetingDate | The Date of the race meeting | DATE | DD-MM-YYYY | - | - | Yes | - | - |
| RaceCourseID\* | The Identifier of the race course | NUMBER(5) | 1-99999 | - | Yes( Race Course) | Yes | - | Index 2 |

**BOOKING TABLE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data type & size** | **Domain(all ranges are inclusive), example or format** | **Primary key** | **Foreign key** | **Required** | **Default value** | **Index** |
| BookingID | The Identifier of the booking | NUMBER(9) | 1-999999999 | Yes | - | Yes | Auto-number | Auto |
| BookingDate | The Date of the booking | DATE | DD-MM-YYYY | - | - | Yes | - | Index 3 |
| Quantity | The quantity of bookings made | NUMBER(1) | 1-8 | - | - | Yes | - | - |
| MeetingID \* | The Identifier of the race meeting | Number(5) | 1-99999 | - | Yes( RaceMeeting) | Yes | - | Index 11 |
| CustomerID\* | The Identifier of the customer | NUMBER(8) | 1-99999999 | - | Yes( Customer) | Yes | - | Index 3 |

**CUSTOMER TABLE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data type & size** | **Domain(all ranges are inclusive), example or format** | **Primary key** | **Foreign key** | **Required** | **Default value** | **Index** |
| CustomerID | The Identifier of the customer | NUMBER(8) | 1-99999999 | Yes |  | Yes | Auto-number | Auto |
| LastName | The Last name of the customer | VARCHAR2(20) | Example : ‘ Freemen ‘ | - |  | Yes | - | Index 4 |
| FirstName | The First name of the customer | VARCHAR2(20) | Example : ‘ Anthony ‘ | - |  | Yes | - | Index 4 |
| StreetAddress | The Street address of the customer | VARCHAR2(40) | Example : ‘ 86 Line Road ‘ | - |  | Yes | - | - |
| Suburb | The Suburb of the customer | VARCHAR2(20) | Example : ‘ Mt Albert ‘ | - |  | Yes | - | - |
| City | The City of the customer | VARCHAR2(20) | Example : ‘ Auckland ‘ | - |  | Yes | - | - |
| PhoneNumber | The Phone number of the customer | VARCHAR2(12) | Example : ‘ 0211144133 ‘ | - |  | Yes | - | - |
| EmailAddress | The email of the customer | VARCHAR2(30) | Example : ‘ tonyfreemen@xtra.co.nz ‘ | - |  | Yes | - | - |
| CreditStatus | The Credit status of the customer | VARCHAR2(7) | V for ‘Valid’ or I for ‘Invalid’ | - |  | Yes | - | - |

**RACE TABLE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data type & size** | **Domain(all ranges are inclusive), example or format** | **Primary key** | **Foreign key** | **Required** | **Default value** | **Index** |
| RaceID | The identifier of the race | NUMBER(9) | 1-999999999 | Yes | - | Yes | Auto-number | Auto |
| RaceName | The name of the race | VARCHAR2(20) | Example : ‘1,500 Metres ‘ | - | - | Yes | - | Index 5 |
| Time | The time for the race | VARCHAR2(6) | 1-999999 | - | - | Yes | - | - |
| Status | The status of the race |  | S for ‘ Scheduled’ or F for ‘Finished’ | - | - | Yes | - | Index 9 |
| RaceTypeDescription | The description of the race type | VARCHAR2() | Example : ‘Flat – Stakes ‘ | - | - | Yes | - | - |
| RaceMeetingID\* | The identifier for the race meeting | NUMBER(5) | 1-99999 | - | Yes( RaceMeeting) | Yes | - | Index 5 |

**ENTRY TABLE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data type & size** | **Domain(all ranges are inclusive), example or format** | **Primary key** | **Foreign**  **Key** | **Required** | **Default value** | **Index** |
| EntryID | The identifier of the entry | NUMBER(9) | 1-999999999 | Yes | - | Yes | Auto-number | Auto |
| RaceID\* | The identifier of the race | Number(9) | 1-999999999 | - | Yes(Race) | Yes | - | Index 6 |
| HorseID\* | The identifier of the horse | NUMBER(5) | 1-99999 | - | Yes(Horse) | Yes | - | Index 6 |
| JockeyID\* | The identifier of the jockey | NUMBER(5) | 1-99999 | - | Yes(Jockey) | Yes | - | Index 6 |
| Status | The status of the entry | VARCHAR2(12) | C for ‘ Confirmed’ or P for ‘Pending’ or D for ‘Disqualified’ | - | - | Yes | - | - |
| HorseTime | The time of the horse | NUMBER(3) | 0-500 | - | - | Yes | - | - |

**HORSE TABLE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data type & size** | **Domain(all ranges are inclusive), example or format** | **Primary key** | **Foreign key** | **Required** | **Default value** | **Index** |
| HorseID | The identifier of the horse | NUMBER(5) | 1-99999 | Yes | - | Yes | Auto-number | Auto |
| HorseName | The name of the horse | VARCHAR2(40) | Example : ‘ Blue Rum ‘ | - | - | Yes | - | Index 7 |
| Gender | The gender of the horse | VARCHAR2(7) | M for ‘Male’ or F for ‘Female’ or G for ‘Gelding’ | - | - | Yes | - | - |
| DateOfBirth | The date of birth of the horse | DATE | DD-MM-YYYY | - | - | Yes | - | - |
| OwnerID\* | The owner of the horse | NUMBER(5) | 1-99999 | - | Yes(Owner) | Yes | - | Index 7 |

**OWNER TABLE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data type & size** | **Domain(all ranges are inclusive), example or format** | **Primary key** | **Foreign key** | **Required** | **Default value** | **Index** |
| OwnerID | The identifier of the owner | NUMBER(5) | 1-99999 | Yes | - | Yes | Auto-number | Auto |
| LastName | The last name of the owner | VARCHAR2(20) | Example : ‘ Page ‘ | - | - | Yes | - | Index 8 |
| FirstName | The first name of the owner | VARCHAR2(20) | Example : ‘ James ‘ | - | - | Yes | - | Index 8 |
| StreetAddress | The street address of the owner | VARCHAR2(40) | Example : ‘ 45 Arney Road ‘ | - | - | Yes | - | - |
| Suburb | The suburb of the owner | VARCHAR2(20) | Example : ‘ Newmarket ‘ | - | - | Yes | - | - |
| City | The city of the owner | VARCHAR2(30) | Example : ‘ Auckland ‘ | - | - | Yes | - | - |
| PhoneNumber | The phone number of the owner | VARCHAR2(12) | Example : ‘ 222 5467 ‘ | - | - | Yes | - | - |
| EmailAddress | The email of the owner | VARCHAR2(30) | Example : ‘ jpage@balham.co.nz ‘ | - | - | Yes | - | - |

**JOCKEY TABLE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data type & size** | **Domain(all ranges are inclusive), example or format** | **Primary key** | **Foreign key** | **Required** | **Default value** | **Index** |
| JockeyID | The identifier of the jockey | NUMBER(5) | 1-99999 | Yes | - | Yes | Auto-number | Auto |
| LastName | The last name of the jockey | VARCHAR2(20) | Example : ‘ White ‘ | - | - | Yes | - | Index 8 |
| FirstName | The first name of the jockey | VARCHAR2(20) | Example : ‘ Jack ’ | - | - | Yes | - | Index 8 |
| Weight | The weight of the jockey | NUMBER(4,2) | 44 - 55 | - | - | Yes | - | - |
| DateOfBirth | The date of birth of the jockey | DATE | DD­ -MM-YYYY | - | - | Yes | - | - |
| EmailAddress | The email address of the jockey | VARCHAR2(30) | Example : ‘ jackwhitextra.co.nz’ | - | - | Yes | - | - |

**INDEXING**

|  |  |  |
| --- | --- | --- |
| **Number** | **Field Name(s)** | **Table Name** |
| 1 | RaceCourseName | RaceCourse |
| 2 | RaceCourseID, MeetingName | Meeting |
| 3 | CustomerID, BookingDate | Booking |
| 4 | LastName. FirstName | Customer |
| 5 | MeetingID, RaceName | Race |
| 7 | HorseName, OwnerID | Horse |
| 8 | LastName, FirstName | Owner |
| 9 | Status, MeetingID | Race |
| 10 | HorseId | Entry |
| 11 | MeetingID | Booking |