

Advanced Database- CA 1

Aleksei Harlasov
X00116756

Table of content

Table of content	i
List of Figures	ii
List of Tables	iii
1 Introduction	1
1.1 Background and Contexts	1
1.2 Database Planning	1
2 Options.....	3
2.1 Approaches.....	3
2.2 Tables	4
2.3 Schema	5
2.4 Query Processing and Optimization.....	5
2.5 Estimate Disk Space Requirements.....	6
2.6 Oracle Data Type	7
References.....	8
Bibliography	9

List of Figures

Figure 1 Database design	2
Figure 2 Tables	5
Figure 3 Schema	5

List of Tables

1 Introduction

1.1 Background and Contexts

The International Basketball Committee (IBC) is required to design and implement a database for Ireland. The database will consist of a functionality based on the Basketball Players Registration System and all data will be retrievable over an Oracle Database system. Since we have a team of Java developers, who don't have a specific knowledge for databases, it will be logical to implement an 'Object-oriented database management system' (OODBMS). But it couldn't be the right solution in relation to service goals. If the correct amount of time isn't taken to map out the project's requirements and define how the database is going to meet them; it is more likely that the whole project will lose direction thus time and money as well. The database will be providing access to players, managers and their clubs. So we can determine 3 tables, which will be described in more details in the "Tables" section.

1.2 Database Planning

Once we are able to determine the purpose of the database, the next step is to check what kind of data is needed to store in the database. For players, we are going to have the following: first name, surname, address with eircode, phone, email, date of birth, club, manager and insurance renewal date. Clubs will consist of just their club name and website.

For managers it's going to be: first name, surname, phone, email, and club. It's very important to understand that planning must be integrated with the overall information system and strategy of the organization. Where it is required to identify what is the goals and plans of the organization, then to evaluate weaknesses and strengths of current information systems.

Determining opportunities or advantages which may exist with new approaches. The purpose of planning is to identify basic business needs and understand how the database will be able to solve problems.

Activities which are associated with the database design are listed below.

Database planning- Planning on how the stages of the lifecycle can be clarified most efficiently and effectively.

System definition- Specifying the scale and boundaries of the database system, including the major user views, its users, and application areas.

Requirements collection and analysis- Collection and analysis of the requirements for the new database system.

Database design- Conceptual, logical, and physical design of the database.

DBMS selection- Selecting a suitable DBMS for the database system.

Application design- Designing the user interface and the application programs that are used and process the database.

Prototyping (optional) - Building a working model of the database system, which allows the designers or users to visualize and evaluate how the final system will look and function.

Implementation- Creating the physical database definitions and the application programs.

Data conversion and loading- Loading data from the old system to the new system and, where possible, converting any existing applications to run on the new database.

Testing- The database system is tested for errors and validated against the requirements specified by the users.

Operational maintenance- Database system is fully implemented. The system is continuously monitored and maintained. When necessary, new requirements are incorporated into the database (Connolly & Begg, 2015).

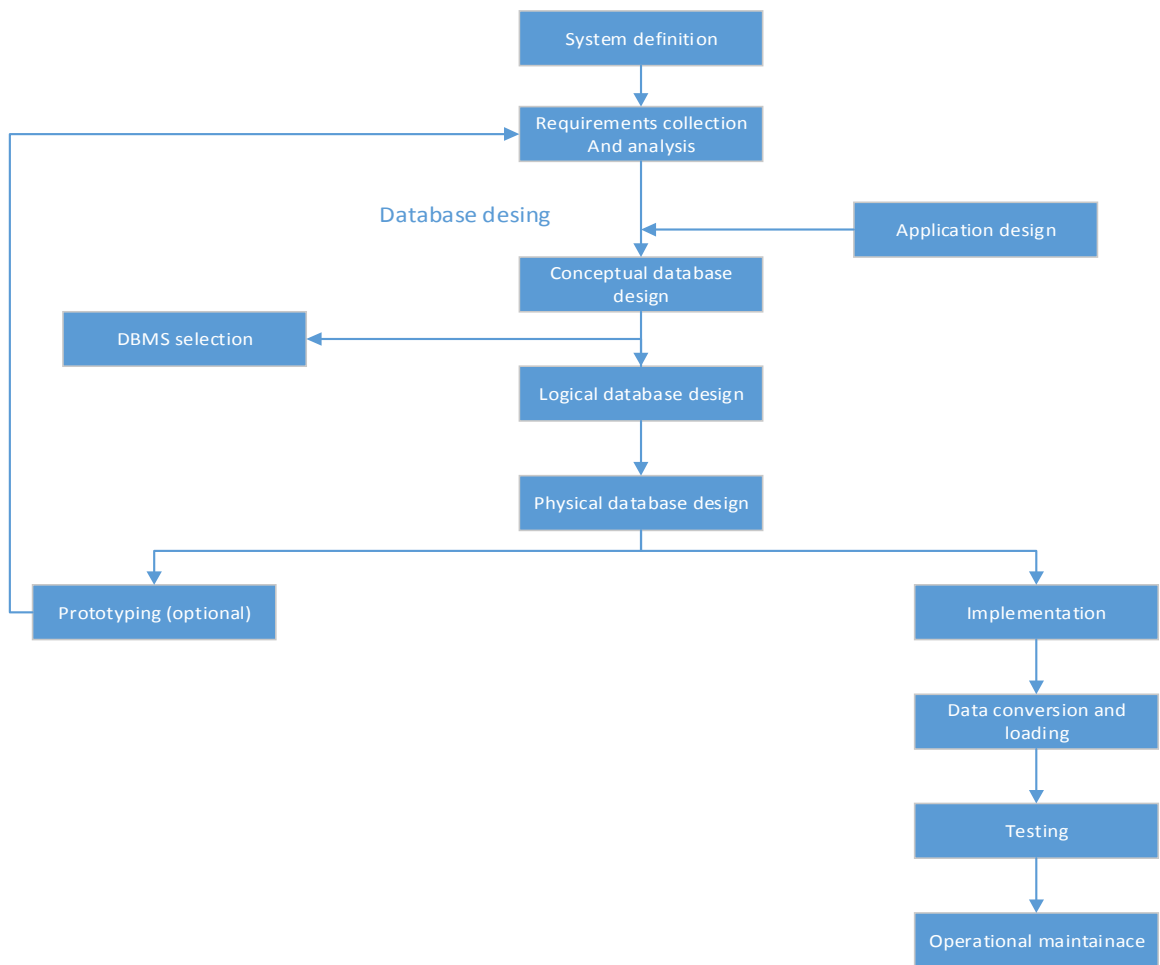


Figure 1 Database design

2 Options

2.1 Approaches

We will consider two main approaches to the design of a database which is initiated as **bottom-top** and **top-down**.

The **bottom-up** approach begins at the fundamental level of attributes (that is, properties of entities and relationships), which through analysis of the associations between attributes are grouped into relations that represent types of entities and relationships between entities. The process of normalization, which represents a bottom-up approach to database design. Normalization involves the identification of the required attributes and their subsequent aggregation into normalized relations based on functional dependencies between the attributes. A more appropriate strategy for the design of complex databases is to use the **top-down** approach. This approach starts with the development of data models that contain a few high-level entities and relationships and then applies successive top-down refinements to identify lower-level entities, relationships, and the associated attributes. The top-down approach is illustrated using the concepts of the Entity-Relationship (ER) model, beginning with the identification of entities and relationships between the entities, which are of interest to the organization (Connolly & Begg, 2015).

Since we are able to identify tables for our database which is “Players”, “Club”, “Managers” we will follow a **top-down** approach. There are other options like **inside-out** approach which is related to the bottom-up approach, but this approach not considering from beginning set of not major entities, relationships and attributes. The **mixed strategy** approach uses both the bottom-top and top-down approach.

The database model is required identification in order to determine the logical structure in which manner data can be store, organised and manipulated.

The process of constructing a model of the data used in an enterprise based on a specific data model, but independent of a particular DBMS and other physical considerations (Connolly & Begg, 2015).

Most popular DBMS model is a relational model which uses a table-based format. The other DBMS-based options, I would like to list below, consist of Object Oriented DBMS, Object Relational DBMS, NoSQL DBMS.

OODBMS- Unlike the relational model, where relationships are not explicitly defined between tables, object oriented models have several ways of specifying relationships between objects directly in the database structure, and these relationships are not implied by the object's contents (values).

Here I would like to add that Object Oriented database management system has no standard language for manipulations and doesn't have compatibility with other standard Application Program Interfaces. The problem is that the data types of the programming languages is different from the attributes data type which is available in data models. Possibly, that could be the reason for developing more flexible DBMS which is Object Related DBMS.

For example, the data types available in C/C++ and Java are different, and both differ from the SQL data types, which are the standard data types for relational databases (Elmasri & Navathe, 2016). Since OODBMS has no strong official support anymore, then this concept is not an option.

ORDBMS- Is a type of DBMS which is some sort of hybrid DBMS. Which is able to provide functionality of OODBMS and RDBMS at least from user perspective.

The fact is, the term “object/relational” is little more than a marketing label, dreamed up to conceal the fact that early so called “relational” products weren't very relational at all (not that most modern ones are either, at least at the time of writing) (Date, 2016).

Object Related DBMS is available for long time but have not been widely adopted compare to RDBMS.

ORDBMS features in the database layer, make the database more complex, and making developers less productive. They have enough headache learning data modeling methods for simple RDBMS, adding structured types and inheritance into their data modeling toolbox just makes it harder (Karwin, 2015).

NoSQL- The term **NOSQL** is generally interpreted as Not Only SQL- rather than NO to SQL- and is meant to convey that many applications need systems other than traditional relational SQL- systems to augment their data management needs (Elmasri & Navathe, 2016).

NoSQL is largely supported by Google, Amazon and Facebook, beside that other software companies developing their own NoSQL systems. Lately NoSQL became very popular, especially on mobile platforms and it's not as restrictive as RDBMS. The advantages of NoSQL is that it has own type of flexibility and simple complexity without too many services like relational SQL. NoSQL corresponding DBMS has different functionalities and strategy compare to relational DBMS. According to documentation we have to focus on implementation of Oracle database system.

RDBMS- A DBMS that manages relational databases (and relational databases only); equivalently, a DBMS that implements the relational model (Date, 2016).

Relational databases or RDBMS, became the norm in IT and powerful enough to make them widely practical and affordable. SQL is widely used in relational databases and has a strong support of DML, which is consist very high level set-oriented DML.

The basic idea is simple and elegant: everything (both entities, like suppliers, and relationships, like shipments of parts by suppliers) is represented in a table (Stajano, 1998).

The reason why RDBMS has retained such a popularity is that its very well established concept, has a strong standards and multiple vendors and best general-purpose solution for data management tasks.

The choice is will be made in a favour of RDBMS implementation into Players Registration System.

2.2 Tables

Tables are the basic unit of data storage in an Oracle database. Database tables hold all user-accessible data. Each table has columns and rows. A table that has an employee database, for example, can have a column called employee number, and each row in that column is an employee's number (Cyran, 2005).

An Oracle database is divided into logical storage units which is representing tablespaces.

The Oracle database consists of three tablespaces to store user data with entities. Business rules define the entities, relationships, and attributes which is based on the database.

The entities are "Players" and "Club" with the relationship. The identifier of PLAYERS as P_ID and the identifier of CLUB as C_ID. The entities are "Managers" and "Club" with the relationship. The identifier of MANAGERS as NAM_ID and the identifier of CLUB as C_ID.

The attributes are listed within each entity.

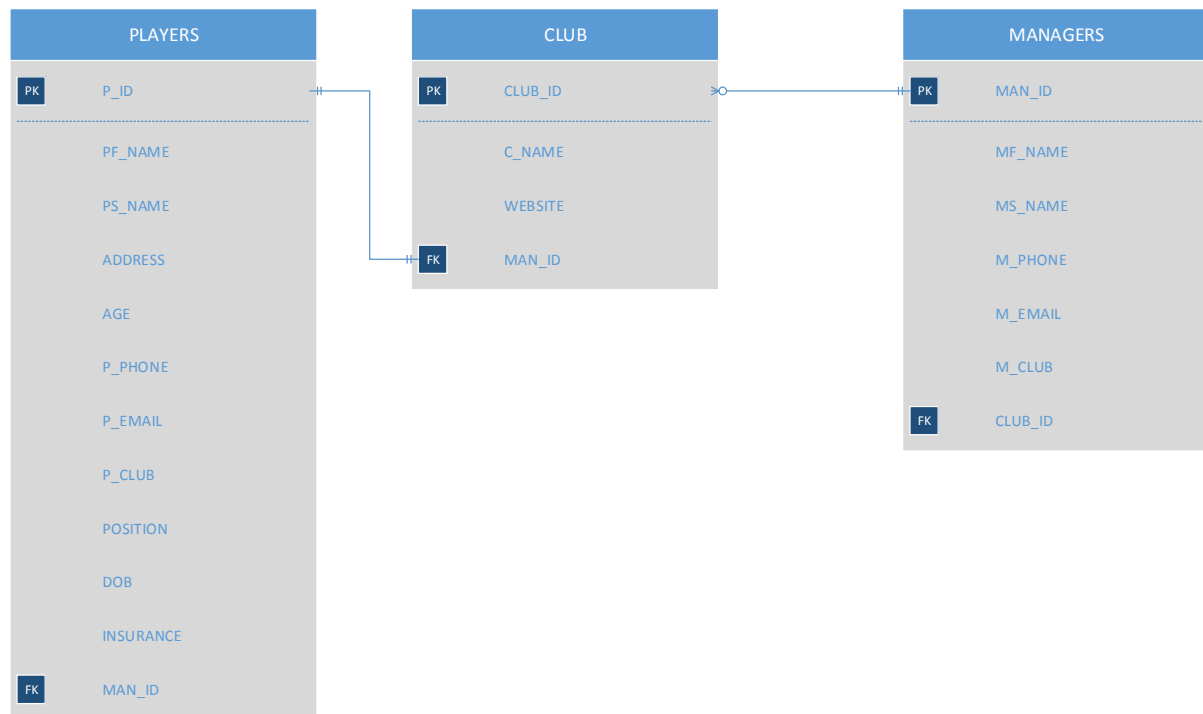


Figure 2 Tables

2.3 Schema

A schema is a collection of database objects. A schema is owned by a database user and has the same name as that user. Schema objects are the logical structures that directly refer to the database's data. Schema objects include structures like structure, views, and indexes. (There is no relationship between a tablespace and a schema. Objects in the same schema can be in different tablespaces, and a tablespace can hold objects from different schemas.) (Cyran, 2005) On this Figure 3 was created three new objects for local schema to support integration of DBMS.

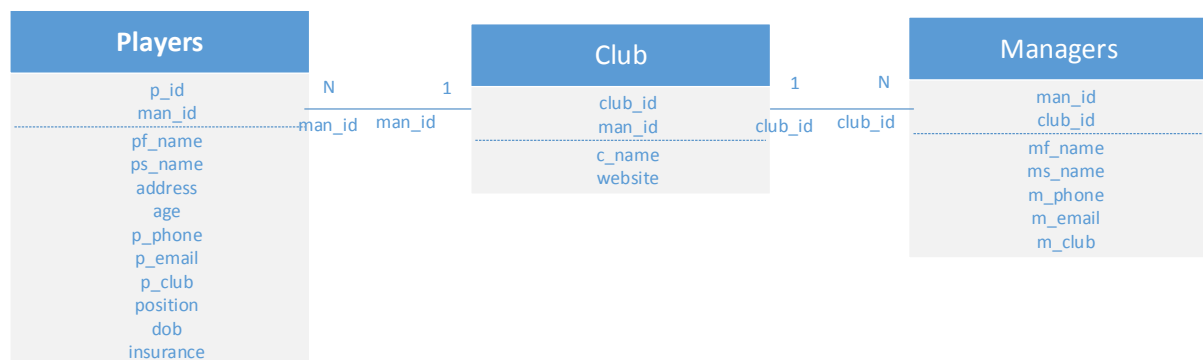


Figure 3 Schema

2.4 Query Processing and Optimization

After the database is created, initialised and populated it needs to be maintained. Various database parameters may need changing and the database may need to be tuned for better performance. Application's data structures may be changed or added, new related application programs may be written to add to the application's functionality.

Optimal Execution Plan is determined based on optimisation plans, available access paths, hints and database analysis. Oracle has a cost-based optimizer that determines join order, join methods, and access paths. Each operation that the optimizer considers has an associated cost

function, and the optimizer tries to generate the combination of operations that has the lowest overall cost (Silberschatz–Korth–Sudarshan, 2004).

It can be adopted a manual optimization or automatic if Oracle Database 10g or higher is considered, where index statistics are automatically gathered when the index is created. If possible a parsing operation can be avoided because it has high cost. The query optimizer going to determine which execution plan is most reasonably efficient by considering available access paths and by factoring in information based on statistics for the schema objects (tables or indexes) in order to be accessed by the SQL statement. To apply efficient solutions for optimizing queries results and low cost election. Oracle DBMS has many options which is based on the optimal path selection. Lack of indexes could lead to a scan of a full table by a function in this case the optimizer cannot use indexes. Specially designed functions which can add an extra flexibility to queries, will consist of joins.

The problem described in Oracle DB system for International Basketball Committee documentation that is some lookups especially on address and date of birth, can take up to minute to return. The possible cause of this problem is that optimizer can use indexes and full table is scanned or tables has a mixed-case data especially when is a large among of data. By analysing these execution plans, it is possible to diagnose the causes of the above problems. Some indexes may be dropped and some new indexes may be created based on the tuning analysis (Elmasri & Navathe, 2016).

2.5 Estimate Disk Space Requirements

It may be one of the steps for planning an implementation of physical database is an estimation of disk space for database. The developing team has to estimate the amount of disk space that is required to store the database. Estimation is highly dependent on the targets of DBMS and the hardware to support the database.

In general, the estimate is based on the size of each tuple and the number of tuples in the relation. The latter estimate should be a maximum number, but it may also be worth considering how the relation will grow and modifying the resulting disk size by this growth factor to determine the potential size of the database in the future (Connolly & Begg, 2015).

First step is to estimate size of table which will consist calculations of clustered index or heap and non-clustered index.

A standard table in Oracle is heap organized; that is, the storage location of a row in a table is not based on the values contained in the row, and is fixed when the row is inserted (Silberschatz–Korth–Sudarshan, 2004).

Where required to specify number of row that will be present in the table:

Num_Rows = number of rows in the table

Specify number of columns and calculate space is required for their storage:

Num_Cols = total number of columns.

Fixed_Data_Size = total of all fixed length columns

Num_Var_Cols = variable-length columns

Max_Var_Size = maximum variable size

Null_Bitmap = $2 + ((\text{Num_Cols} + 7) / 8)$ = null bitmap

Variable_Data_Size = $2 + (\text{Num_Variable_Cols} \times 2) + \text{Max_Var_Size}$ = calculate variable-length columns in the table

Row_Size = $\text{Fixed_Data_Size} + \text{Variable_Data_Size} + \text{Null_Bitmap} + 4$ = calculate total row size

Rows_Per_Page = $8096 / (\text{Row_Size} + 2)$ = calculate number of rows per page

Num_Pages = $\text{Num_Rows} / \text{Rows_Per_Page}$ = number of pages to store all the rows

Heap size (bytes) = 8192 x *Num_Pages* = *Calculate the amount of space that is required to store the data in the heap*

2.6 Oracle Data Type

Each column value and constant in a SQL statement has a datatype, which is associated with a specific storage format, constraints, and a valid range of values. When table is created, the next step is to specify a datatype for each of its columns.

Oracle provides the following built-in datatypes:

Character datatypes - VARCHAR2, string length is between 1 and 4000 bytes or characters.

Numeric datatypes – NUMBER stores fixed and floating point numbers, up to 39 digits.

Date datatype – DATE stores date and time values.

CREATE TABLE PLAYERS

```
(
P_ID          NUMBER(5) NOT NULL,
PF_NAME       VARCHAR2(40) NOT NULL,
PS_NAME       VARCHAR2(40) NOT NULL,
ADDRESS       VARCHAR2(40) NOT NULL,
AGE           NUMBER(4) NOT NULL,
P_PHONE       NUMBER (40) NOT NULL,
P_EMAIL       VARCHAR2(40) NOT NULL,
P_CLUB        VARCHAR2(40) NOT NULL,
POSITION      VARCHAR2(40) NOT NULL,
DOB           DATE,
INSURANCE     VARCHAR2(40) NOT NULL,
PRIMARY KEY (p_id),
CONSTRAIN    man_id REFERENCES club (club_id)
);
```

References

- Connolly, T. & Begg, C., 2015. *Database Systems: A Practical Approach to Design, Implementation, and Management*. 6th ed. Harlow: s.n.
- Cyran, M., 2005. *Oracle Database Concepts*. s.l.:s.n.
- Date, C., 2016. *The New Relational Database Dictionary*. Sebastopol, CA: O'Reilly Media, Inc.
- Elmasri, R. & Navathe, S., 2016. *Fundamentals of Database Systems*. 7th ed. NJ: Pearson.
- Jones, A., 2015. *Cloud Computing*. 1st ed. San Francisco: Johnson.
- Karwin, B., 2015. *Are object-related database the future?*. [Online] Available at: <https://www.quora.com/Are-Object-relational-databases-the-future> [Accessed 18 november 2016].
- Silberschatz–Korth–Sudarshan, 2004. *Database System Concepts, 4th Ed*, s.l.: Foxit Software Company.
- Stajano, F., 1998. *A Gentle Introduction to Relational and Object Oriented Databases*. [Online] Available at: <https://www.cl.cam.ac.uk/~fms27/db/tr-98-2.pdf> [Accessed 18 november 2016].

Bibliography

FUNDAMENTALS OF Database Systems 7th Ed, Ramez Elmasri, Shamkant B. Navathe

Database design: know it all / Toby Teorey et al. p. cm. — (Morgan Kaufmann know it all series). 2009.

Databases for Small Business: Essentials of Database Management, Data Analysis, and Staff Training for Entrepreneurs and Professionals, Anna Manning, 2015.

Database Systems, A Practical Approach to Design, Implementation, and Management 6th Ed, Thomas M. Connolly z Carolyn E. Begg, 2015.

Principles of Distributed, Database Systems, 3rd, M. Tamer Özsu, Patrick Valduriez, 2015.

Fundamentals of Database Management System, 2nd Ed, MARK L. GILLENSON, 2015.

Modern Database Management - 10th Ed, Jeffrey Hoffer, V Ramesh, Heikki Topi, 2012.

The New Relational Database Dictionary, C. J. Date, 2015.