Outline

Data Dictionary storage in DBMS

Data Dictionary

A Data Dictionary comprises two words i.e. **Data** which simply means information being collected through some sources and **Dictionary** means where this information is available.
A **Data Dictionary** can be defined as a collection of information on all data elements or contents of databases such as data types, and text descriptions of the system. It makes it easier for users and analysts to use data as well as understand and have common knowledge about inputs, outputs, components of a database, and intermediate calculations.

Why Data Dictionary is Essential?

There is less information and details provided by data models. So, a data dictionary is essential and needed to have proper knowledge and usage of contents. Data Dictionary provides all information about names that are used in system models. Data Dictionary also provides information about entities, relationships, and attributes that are present in the system model. As a part of the structured analysis and design tool, the implementation of a data dictionary is done.

The following data name type of Information is used to store in a data dictionary:

Why Data Dictionary is Essential?

Name	Description
Name	Name generally includes the primary name of all composite data or control items available, and the name of the external entity or data store.
Aliases	Any other word used in place of Name
Where or How it’s used?	A data dictionary generally gives information about where and how data or control items are used which may include an input/ output to process.
Description	A notation for representing content

Types of Data Dictionary

Data Dictionary is basically of two types. We will discuss each of them.

- Integrated Data Dictionary
- Stand Alone Data Dictionary

1. Integrated Data Dictionary

Integrated Data Dictionary can be seen as a catalog that can be maintained by the relational database. In previous databases, there is not any functionality of Integrated Data Dictionary, so they use Stand Alone Data Dictionary.

There are two types of **Integrated Data Dictionary.**

- **Active Data Dictionary:** Active Database Dictionary is a type of database that is updated automatically in case any changes are to be done to the database. These are self-updating.
- **Passive Data Dictionary:** Passive Databases are the databases that have to be maintained or updated manually in case of any changes have been made to the system.

Types of Data Dictionary

2. Stand Alone Data Dictionary

Stand Alone Data Dictionary is a type of flexible data dictionary as Database Administrator has ease of managing data. It does not require data that is computer-based. It has no fixed format. But some elements are common in this kind of database.

•**Data Elements:** It has the elements like name, datatype, validation rules, etc.

•**Tables:** These contain all the necessary information that is required for the table, how many rows in the table, how many columns in the table, etc.

•**Index:** The index of the databases is to be stored here.

•**Programs:** These are used for accessing the database, and can include SQL Queries, Reports, etc.

•**Relationship between Data Elements:** This stores the relationship among the different databases, like cardinality, connectivity, etc.

How to Create a Data Dictionary?

Data Dictionary can be formed with the help of the following relational database like MySQL, SQL Server, etc. Several notations are present everywhere which enhances the quality of the Data Dictionary. We will also look into that. Database Administrators can have the following templates like SQL Server, that help in making Data Dictionary.

Notations in Data Dictionary

Data Construct	Notation	Meaning
Composition	=	Is composed of
Sequence	+	Represents AND
Selection	[|]	Represents OR
Repetition	$\{\ \}^n$	Represents n repetitions or repetition for n times
Parentheses	( )	Optional data that may or may not be present
Comment	*…*	Delimits a comment or commented information

How to Create a Data Dictionary?

Example: Let us understand this by taking an example of the reservation system. In the reservation system, “passenger” is a data item whose information is available on the data dictionary as follows:

Keys	Values
Name	Passenger
Aliases	None
Where or how it’s used?	Passenger’s query (input) Ticket (output)
Description	•Passenger = Passenger_name + Passenger_address •Passenger_name = Passenger_lastname + Passenger_firstname + Passenger_middle_initial •Passenger_address = Local_address + Community_address + Zip_code •Local_address = House_number + street_name + Apartment_number •Community_Address = City_name + State_name

Challenges with Data Dictionary

- The main challenge that occurs in front of us is that a data dictionary is somehow difficult and it may lead to take much time in case when data is not prepared previously.
- Data Preparation is a bit lengthy and hectic process for a large scale of data.
- When you don’t do Data Preparation, Data Dictionary is not that much efficient.
- It can be an expensive process when resources are not to be utilized efficiently.
- Data Dictionary can be an insecure process as if you give access to so many persons, it might be a challenge to the security of the Data Dictionary.

Buffer Manager

Programs call on the buffer manager when they need a block from disk.

1. If the block is already in the buffer, buffer manager returns the address of the block in main memory

2. If the block is not in the buffer, the buffer manager

    1. Allocates space in the buffer for the block

        1. Replacing (throwing out) some other block, if required, to make space for the new block.

        2. Replaced block written back to disk only if it was modified since the most recent time that it was written to/fetched from the disk.

    2. Reads the block from the disk to the buffer, and returns the address of the block in main memory to requester.

Buffer-Replacement Policies

Most operating systems replace the block least recently used (LRU strategy). Idea behind LRU – use past pattern of block references as a predictor of future references.Queries have well-defined access patterns (such as sequential scans), and a database system can use the information in a user’s query to predict future references.

LRU can be a bad strategy for certain access patterns involving repeated scans of data

⍰ For example: when computing the join of 2 relations r and s by a nested loops

```
        for each tuple tr of r do
       for each tuple ts of s do
if the tuples tr and ts match …
```

Mixed strategy with hints on replacement strategy providedby the query optimizer is preferable

Buffer-Replacement Policies (Cont.)

- ❑Pinned block – memory block that is not allowed to be written back to disk.
- ❑Toss-immediate strategy – frees the space occupied by a block as soon as the final tuple of that block has been processed
- ❑Most recently used (MRU) strategy – system must pin the block currently being processed. After the final tuple of that block has been processed, the block is unpinned, and it becomes the most recently used block.
- ❑Buffer manager can use statistical information regarding the probability that a request will reference a particular relation

E.g., the data dictionary is frequently accessed. Heuristic: keep data-dictionary blocks in main memory buffer

- ❑Buffer managers also support forced output of blocks for the purpose of recovery

Thank You