# DATABASE SCHEMA COMPLEXITY REDUCTION

## BACKGROUND OF THE INVENTION

### 1. Field of the Invention

The invention relates to the field of database design and data storage, representation or manipulation.

### 2. Description of the Related Art

A historically recurring issue with large databases is that describing the structural relationships between multiple data items has required a database structure which is correspondingly large, complex and unwieldy. Maintaining such a complex structure has proven problematic, resulting in over-long access times and increased complexity when extracting or updating the data in such databases.

In addition, the historical solution – which has relied on the creation of large numbers of cross-referencing database tables – has also meant that extending the database's scope to incorporate further types of metadata has required that additional cross-reference tables be added to the database schema, further increasing its complexity and requiring major alterations to the code used to access that database.

The present invention discloses a novel method for producing database schemas which resolve those historical problems.

## BRIEF SUMMARY OF THE INVENTION

In contrast to the traditional approach of defining relationships between data collections using complex groups of cross-reference database tables, the present invention makes use of generic sets to represent relationships between complex collections of interrelated data. The complexity of a database schema can be greatly reduced by representing the relationships between data items as these data sets (sets of data items). These sets of data items may be defined in one or more database tables in a database.

For example, in the context of a database that stores a digital media catalogue (e.g. several tens of thousands of MP3 music tracks), then there could a Track Database Table; each individual track occupies a row in the table, with accompanying metadata in that row. A Track Set Table then lists all the different sets, where a set is a collection of tracks with a common characteristic – one set could be classical tracks, another set could be 'most played' tracks, another set could be favorite driving tracks (sets can be automatically generated or defined manually or by the use of a tool). So each row in this Track Set Table lists a particular set, plus accompanying metadata and a unique ID to aid retrieval of that particular set. Then, we have a third table, the Track Set Item table; each row in this table lists the associated unique ID for the set that the track belongs to plus additional metadata such as the minimum information needed to be displayed to a user (e.g. artists name; track name), plus a reference to the full data (e.g. MP3 file) in the Track Database Table. This table structure in effect separates the 'core' data (e.g. the several thousand MP3 tracks) from the relationships between the tracks and allows a much simplified database structure or schema.

This approach provides for much simplified database data management and control. It enables the easy pre- or re- calculation of sets and provides for the easy creation of hierarchical meta-data trees, which are ideal for the targeted delivery of data to the user.

## Definitions

For convenience, and to avoid needless repetition, the terms "music" and "media content" in this document are to be taken to encompass all "media content" which is in digital form or which it is possible to convert to digital form - including but not limited to books, magazines, newspapers and other periodicals, video in the form of digital video, motion pictures, television shows (as series, as seasons and as individual episodes), computer games and other interactive media, images (photographic or otherwise) and music.

Similarly, the term "track" indicates a specific item of media content, whether that be a song, a television show, an eBook or portion thereof, a computer game or any other discreet item of media content.

The terms "playlist" and "album" are used interchangeably to indicate collections of "tracks" which have been conjoined together such that they may be treated as a single entity for the purposes of analysis or recommendation.

The terms "digital media catalogue", "digital music catalogue", "media catalogue" and "catalogue" are used interchangeably to indicate a collection of tracks and/or albums to which a user may be allowed access for listening purposes.

The abbreviation "DRM" is used to refer to a "Digital Rights Management" system or mechanism used to grant access rights to a digital media file.

The verb "to listen" is to be taken as encompassing any interaction between a human and media content, whether that be listening to audio content, watching video or image content, reading books or other textual content, playing a computer game, interacting with interactive media content or some combination of such activities.

The terms "user", "consumer", "end user" and "individual" are used interchangeably to refer to the person, or group of people, whose media content "listening" preferences are analysed and for whom recommendations are made. In all cases, the masculine includes the feminine and vice versa.

The terms "device" and "media player" are used interchangeably to refer to any computational device which is capable of playing digital media content, including but not limited to MP3 players, television sets, home computer systems, mobile computing

devices, games consoles, handheld games consoles, vehicular-based media players or any other applicable device or software media player on such a device.

The "server" is a computing device which operates as a database server.

The "client" device is a computing device and which may be a mobile computing device. The client device may be the same computing device as the server in some configurations.

The "remote database" consists of a collection of data items, however represented, on the server.

The "local database" consists of a collection of data items, however represented, on the client device.

The "network" is a method of communicating between the said server and the said client device using, for example, the internet, a local network or a wireless network.

In the preferred embodiment, the database and data items referred to are structured in a relational database which is accessed, directly or indirectly, using Structured Query Language (SQL) or some equivalently powerful Application Programming Interface (API). In the preferred embodiment, the local database is provided via an embedded C database engine. In another embodiment, the client application utilises an external database, whether external to the client application software or external to the client device entirely.

## DETAILED DESCRIPTION

In contrast to the traditional approach of defining relationships between data collections using complex groups of cross-reference database tables, the present invention makes use of generic sets to represent relationships between complex collections of interrelated data.

This method provides for much simplified database data management and control. The database may (but does not have to be) a relational database. It enables the easy pre- or re- calculation of sets and provides for the easy creation of hierarchical meta-data trees which are ideal for the targeted delivery of data to the user.

### 1. Defining Sets

In the preferred embodiment, sets are defined in the database using a pair of tables, one defining the set itself and another defining the items in that set. In another example embodiment the two tables are collapsed into a single database table while other example embodiments represent the sets-"set item" relationship using more than two database tables but nonetheless preserve the essential structure of representing a set-to-"set item" relationship.

The "Set" table stores such metadata as is required to retrieve a specific set or group of sets. In the preferred embodiment, that metadata includes fields which are used to store the type of a set, such as that an "artist set" is a set of artists which are all related to a particular "genre" of music.

The "Set Item" table stores individual items with a set, with each item being linked to its enclosing set using any reasonable method, including the use of traditional "foreign keys" into the "Set" database table.

Each row in the "Set Item" table may, in one example embodiment, include a foreign key into another database table which contains the actual data for that item. For example, a set item which describes an Artist may have a foreign key linking into an "Artist" database table which may be used to retrieve the full data for the artist to which that "set item" refers. In another example embodiment, the "set item" rows may duplicate some or all of the data in that "Artist" table, in order to permit more immediate access to that data (i.e. removing the need to link to the separate Artist table when querying the

database). More generally, items in the data sets may include de-normalised associated information.

In yet another example embodiment, the "set item" table and the said "Artist" table are in fact the same database table. In the preferred embodiment, a combination of the said techniques is employed, in that the "set item" table rows contain both a foreign key in an "Artist" table and duplicate such data from that "Artist" table in the "Set Item" table as is needed when retrieving items for a given set.

By separating the core data from the relationships between data items using the method disclosed herein, the database structure is able to be considerably simplified from the traditional model, which makes use of multiple cross-reference tables to retrieve those collections of data which are actually used in the computer program which is utilised to access the database.

For performance and simplicity of use, de-normalisation is built in to the database design in the preferred embodiment. Hence often a name will accompany a foreign key for speed of reference to the name which is often, with the ID, the only thing of interest required initially when a computer program requests a set of data.

**2. Creating sets**

In addition, the data sets themselves may be, and in the preferred embodiment are, pre-defined via pre-processing of data or manually using a software tool or dynamically through software as data changes or events occur. In the preferred embodiment, any combination of some or all of the aforementioned processes may be used. The type of data set may be wholly or partially defined by the database table used to store that data set.

In one example embodiment, data representing music tracks is to be loaded into the database. Prior to loading the said data, it is passed through a pre-processing tool which identifies the genre of each track – either by examining the metadata associated with each track or by analysing the track's audio using a DSP (Digital Sound Processing) algorithm or by manual identification of the track genre or by some combination of one or more of the said techniques.

The pre-processing tool, in the example embodiment presented, assigns each track to one or more "track sets" based (for example) on the genre associated with that track.

This is done by creating a row for that genre in the Track Set table, if no such row exists, and then adding a row for that track in the Track Set Item table and linking the said row to the associated Track Set.

In another example embodiment, the creation of sets is performed simultaneously with loading the data into the database. In still another embodiment, sets may be created based on data already loaded into the database.

In the said example embodiment, tracks contain associated metadata of arbitrary size and complexity, some or all of which is loaded into a "Track" database table. In this example embodiment, a "Track Set" database table is designed to be utilised by a computer program which requires only a small subset of that metadata when retrieving a set of tracks. In the preferred embodiment, the set definition tool copies that subset of data such that that metadata is duplicated in the "Track Set Item" database table, for ease of access.

In the preferred embodiment, sets are defined in one or more database tables, where the storage table chosen to store a particular data set may be considered to be metadata about the type of set. For example, a database table **track_set** may contain sets of music tracks while a database table **artist_set** contains sets of artists.

In the preferred embodiment, sets may be defined where the "set item"s for that set are themselves sets. This provides for the creation of sets-of-sets, which may be hierarchical in nature where required.

### 3. Hierarchical meta-data trees

The methods disclosed by the present invention provide for the easy creation of hierarchical meta-data trees which are ideal for the targeted delivery of data to the user.

By creating sets of sets, whereby the "set items" which constitute the elements of a set are themselves sets, the preferred embodiment of the present invention allows for the definition and usage of collections of sets which may, where required, be hierarchical in nature.

### 4. Retrieving Sets

Sets are identified in the Track Set table using metadata which permits a given set (which may be, in the preferred embodiment, a set of sets) to be retrieved. For example, a query

for a set of music tracks with the genre "Light Classical" would retrieve a set from that "Track Set" table where it is a track set of type "genre" and with the sub-type of "Light Classical" using a fast and computationally efficient table look-up process. The collection of Track Set Items linked to that set is then, in the preferred embodiment, returned to the requesting computer program.

In the preferred embodiment, specific sets can be referenced from anywhere in the system. Hence new types of lists may be added easily without having to build new cross-reference tables etc. Rather, the system need only reference the foreign key (the "Set ID") of the set type of choice. The application may then retrieve any specific set using that "Set ID" as a foreign key into the table of sets.

**5. Referential Integrity**

In the preferred embodiment, the database is fully RI (Referential Integrity) protected through triggers, constraints, functions or procedures:

- Cardinality and existence rules on Foreign Key relationships (cascade, restrict on update/delete operations)
- Triggers for pre/post insert/update/delete as required.
- Default values, NULL settings and Constraint checks on fields

The RI protection in the preferred embodiment serves one or more of the following functions:

- Modified and Inserted fields are completely handled in the database and should not be set or modified by the application.
- ID fields are completely handled in the database and should not be set or modified by the application. Hence, in the preferred embodiment, if you insert, then you need to read the inserted record (object) afterwards to have a complete copy of the inserted row (object) containing the newly allocated ID.
- Count fields throughout the database
- Perform field and record-field validation of the possible allowed combination of fields. Such validation conditions are, in the preferred embodiment, checked in the database using triggers in order to completely protect its integrity.

In the preferred embodiment, the referential integrity checks are used to, at a minimum, ensure that (1) the insertion of a "Set Item" is permitted only if the associated Set exists and that (2) the deletion of a set is permitted either only if that set has no associated "set items" rows. In another example embodiment, the deletion of a Set triggers the automatic deletion of that Set's associated "Set Item" rows.

## CLAIMS

1. A method for reducing the complexity of a database schema by representing the relationships between data items as "data sets" (sets of data items).

2. The method in 1 where the sets of data items are defined in one or more database tables in a database

3. The method of Claim 2, in which the database table(s) represent a set-to-"set item" relationship.

4. The method of any preceding Claim 2 or 3 in which a table stores such metadata as is required to retrieve a specific set or group of sets.

5. The method of any preceding Claim where the data sets are defined manually.

6. The method of any preceding Claim 1 – 4 where the data sets are defined using a pre-processing tool or script

7. The method of any preceding Claim 1 – 4 where the data sets are defined dynamically by software as data changes or user actions occur

8. The method of any preceding Claim where the items in the data sets include some de-normalised associated information

9. The method of any preceding Claim where the database's referential integrity is protected using zero, one or more triggers, constraints, functions and procedures.

10. The method of any preceding Claim where one or more data sets can be retrieved using metadata associated with data sets, including but not limited to the type of set.

11. The method in 10 where the type of data set is wholly or partially defined by the database table used to store that data set.

12. The method of any preceding Claim where a unique set identifier is defined for each data set and may be used to retrieve that specific data set.

13. The method of any preceding Claim where the said "data items" are themselves data sets.

14. The method in 13 where the said data sets are defined so as to form a hierarchy of data sets.

15. The method of any preceding Claim where the database is a database of digital media content.