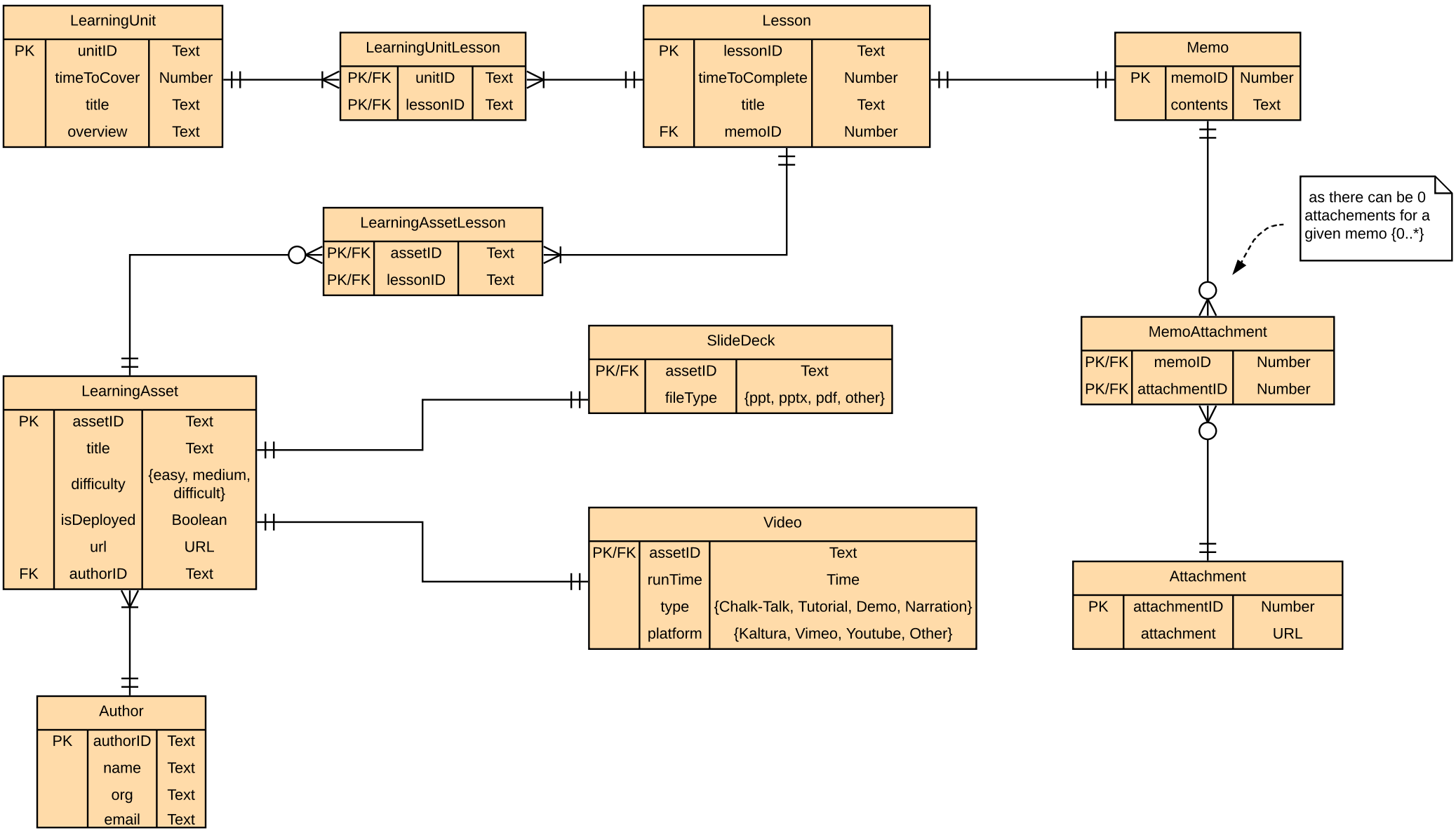
ERD (Logical data model) in the IE/Crow's Foot Notation



Relational Schema Definitions

LEARNING_UNIT { unitID, time_to_cover, title, overview }

LESSON { lessonID, time_to_complete, title, *memoID* }

LEARNING_UNIT_LESSON { unitID, lessonID }

MEMO { memoID, contents }

ATTACHMENT { attachmentID, attachment }

MEMO_ATTACHMENT { memoID, attachmentID }

LEARNING_ASSET { assetID, title, difficulty, is_deployed, url, *authorID* }

LEARNING_ASSET_LESSON { assetID, lessonID }

SLIDE_DECK { assetID, file_type }

VIDEO { assetID, run_time, type, platform }

AUTHOR { authorID, name, org, email}

Reasoning and Assumptions :

1) Relational models don't allow implementation of direct many-to-many relationships between two tables because it is not possible to store the data efficiently. Hence, for efficient processing, we need to convert many-to-many relationship tables into two "one-to-many" relationships by connecting these two tables with an intersection/linking table that contains the keys of both these tables. Hence, I have the tables "LearningUnitLesson", "LearningAssetLesson" and "MemoAttachment" as linking tables.

2) The overview attribute which is given as a derived attribute in the question is represented normally in the ERD.

3) Since both SlideDeck and Video entities have the attribute "url", I have included it in its parent class LearningAsset (generalization)

4) Since "attachment" in Memo is a multi-valued attribute, I have decomposed/normalized it to form a new table "Attachment", this ensures that all the attributes are single-valued.

5) While the given UML diagram contains Aggregation and Generalization, an ERD for a relational database does not and hence I have not represented them as there is no visual notation for it. Instead I have used multiplicity to depict the same. However, these concepts cannot be achieved to their true and complete meaning in this way.

6) The Generalisation of LearningAsset - which is an **abstract superclass** (as LearningAsset cannot exist on its own without SlideDeck or Video) has been represented in the ERD as a **1 to 1** relationship between the superclass and its subclasses since LearningAsset **has to be** of the type SlideDeck or Video. However, as expected this leads to implementation problems where a LearningAsset can be both SlideDeck and Video, but this should not be the case since the generalisation is **disjoint** (so the primary key of the subclasses will be different), this issue can be solved at the application level using triggers. *(Note: the concept of generalisation of an abstract superclass has been taught in the tutorial - "Patterns for mappling models to schemas" as a part of the course modules)*