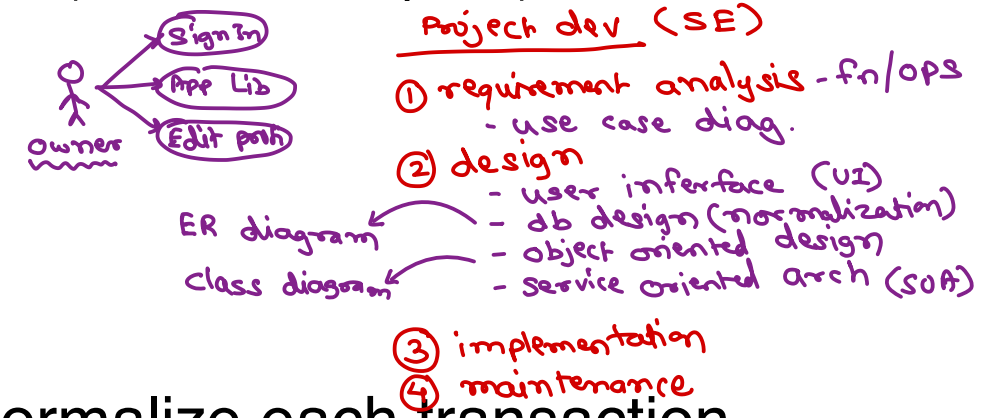
MySQL RDBMS

Trainer: Mr. Nilesh Ghule

Normalization

- Concept of table design: Table, Structure, Data Types, Width, Constraints, Relations.
- Goals:
  - Efficient table structure.
  - Avoid data redundancy i.e. unnecessary duplication of data (to save disk space).
  - Reduce problems of insert, update & delete.
- Done from input perspective.
- Based on user requirements.
- Part of software design phase.
- View entire appln on per transaction basis & then normalize each transaction separately.
- Transaction Examples:
  - Banking, Rail Reservation, Online Shopping.



Normalization

- For given transaction make list of all the fields.
- Strive for atomicity.
- Get general description of all field properties.
- For all practical purposes we can have a single table with all the columns. Give meaningful names to the table.
- Assign datatypes and widths to all columns on the basis of general desc of fields properties.
- Remove computed columns.
- Assign primary key to the table.
- At this stage data is in un-normalized form.
- UNF is starting point of normalization.

Normalization

- UNF suffers from
  - Insert anomaly → some data need to be inserted repeatedly
  - Update anomaly → same changes might be done at multiple records.
  - Delete anomaly → some of undesired data may be deleted.

1-NF → 25%.
steps 1-3

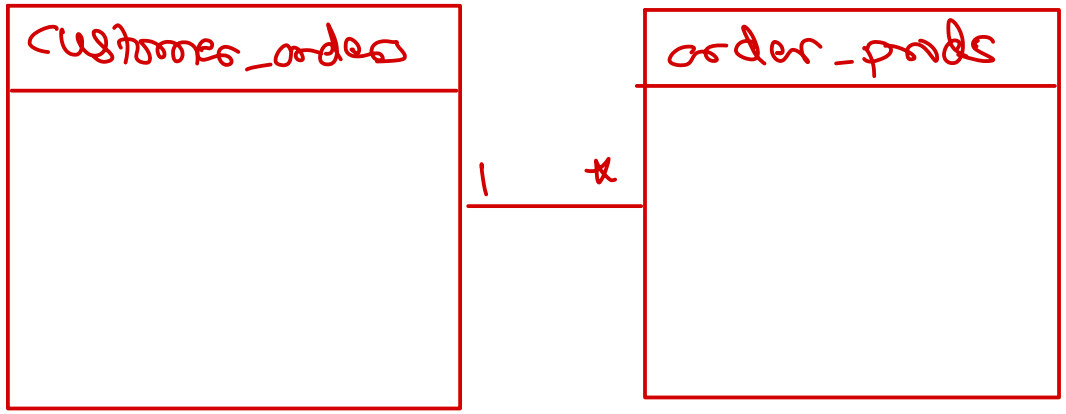
2-NF → 50%.
steps 4-6

3-NF → 25%.
steps 7-9

BCNF → optional
steps 10

Normalization 1-NF

- 1. Remove repeating group into a new table.
- 2. Key elements will be PK of new table.
- 3. (Optional) Add PK of original table to new table to give us Composite PK.
  - Repeat steps 1-3 infinitely -- to remove all repeating groups into new tables.
  - This is **1-NF**. No repeating groups present here. One to Many relationship between two tables.



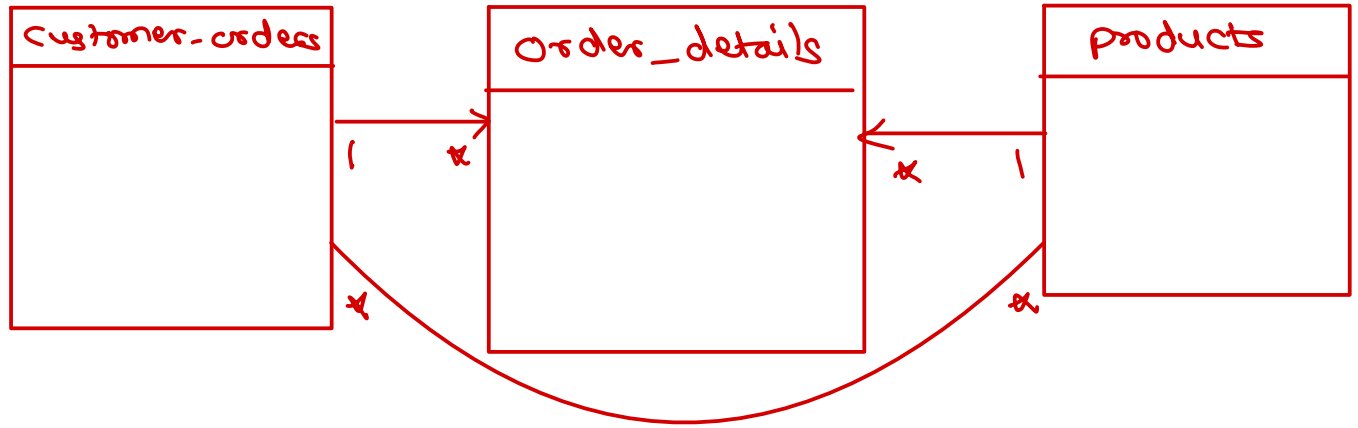
Normalization - 2 NF

- 4. Only table with composite PK to be examined.
- 5. Those columns that are not dependent on the entire composite PK, they are to be removed into a new table.
- 6. The key elements on which the non-key elements were originally dependent, it is to be added to the new table, and it will be the PK of new table.
  - Repeat steps 4-6 infinitely -- to separate all non-key elements from all tables with composite primary key.
  - This is **2-NF**. Many-to-Many relationship.

Many to many

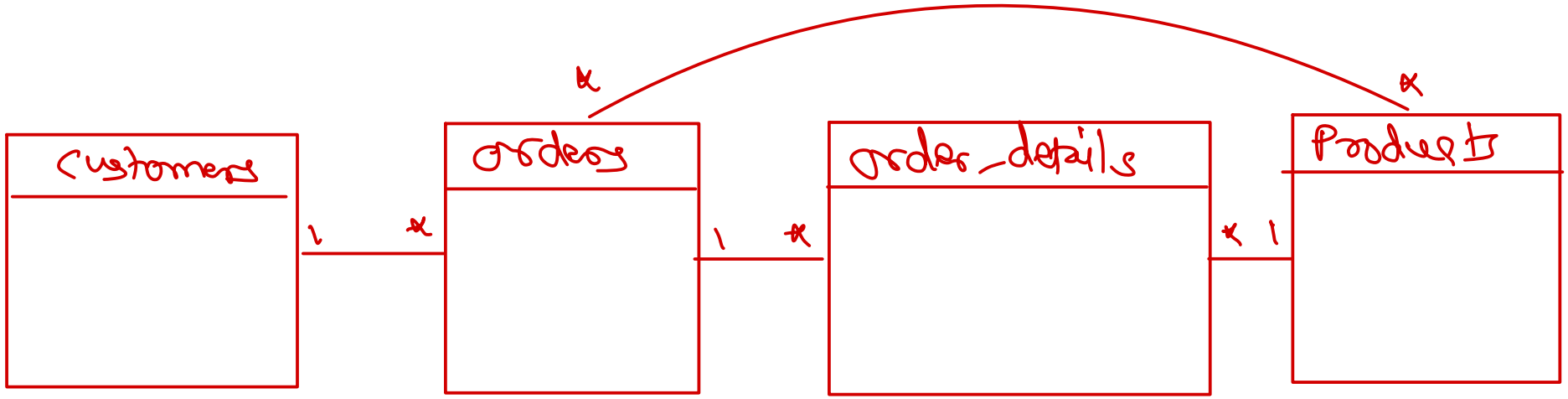
One order has many products.

One product is purchased in many orders.



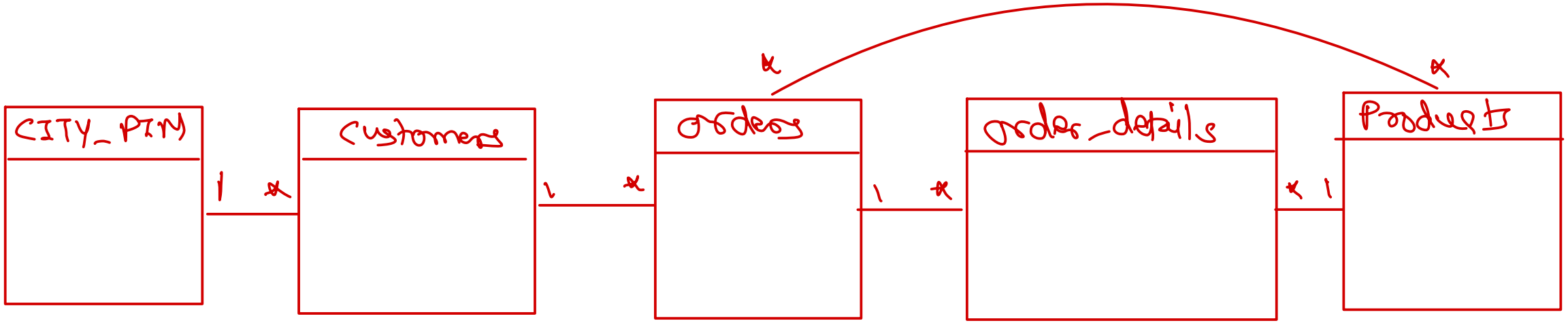
Normalization

- 7. Only non-key elements are examined for inter-dependencies.
- 8. Inter-dependent cols that are not directly related to PK, they are to be removed into a new table.
- 9. (a) Key ele will be PK of new table.
- 9. (b) The PK of new table is to be retained in original table for relationship purposes.
  - Repeat steps 7-9 infinitely to examine all non-key eles from all tables and separate them into new table if not dependent on PK.
  - This is **3-NF**.



Normalization

- To ensure data consistency (no wrong data entered by end user).
- Separate table to be created of well-known data. So that min data will be entered by the end user.
- This is BCNF or 4-NF.



SQL Keys

🔑 **An SQL key is either a single column (or attribute) or a group of columns that can uniquely identify rows (or tuples) in a table.**

🔑 **Super key is a single key or a group of multiple keys that can uniquely identify tuples in a table.**

🔑 **Candidate key is a single key or a group of multiple keys that uniquely identify rows in a table.**

🔑 **Primary key is the Candidate key selected by the database administrator to uniquely identify tuples in a table.**

🔑 **Alternate keys are those candidate keys which are not the Primary key.**

🔑 **Foreign key is an attribute which is a Primary key in its parent table, but is included as an attribute in another host table.**

Id	Name	Gender	City	Email	Dep_Id
1	Ajay	M	Delhi	ajay@gmail.com	1
2	Vijay	M	Mumbai	vijay@gmail.com	2
3	Radhika	F	Bhopal	radhika@gmail.com	1
4	Shikha	F	Jaipur	shikha@gmail.com	2
5	Hritik	M	Jaipur	hritik@gmail.com	2

5 rows in set (0.00 sec)

De-normalization

- Normalization will yield a structure that is non-redundant.
- Having too many inter-related tables will lead to complex and inefficient queries.
- To ensure better performance of analytical queries, few rules of normalization can be compromised.
- This process is de-normalization.

Codd's rules

- Proposed by Edgar F. Codd – pioneer of the RDBMS – in 1980.
- If any DBMS follow these rules, it can be considered as RDBMS.

- The 0th rule is the main rule known as "The foundation rule".
  - For any system that is advertised as, or claimed to be, a relational data base management system, that system must be able to manage data bases entirely through its relational capabilities.

- The rest of rules can be considered as elaboration of this foundation rule.

Codd's rules

- Rule 1: The information rule:
  - All information in a relational data base is represented explicitly at the logical level and in exactly one way – by values in tables.

- Rule 2: The guaranteed access rule:
  - Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name.

- Rule 3: Systematic treatment of null values:
  - Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type.

Codd's rules

metadata/structure/ psm

- Rule 4: Dynamic online catalog based on the relational model:
  - The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.

MySQL 5.0+ → db information_schema.
TRIGGERS ROUTINES ...

- Rule 5: The comprehensive data sublanguage rule:
  - A relational system may support several languages. However, there must be at least one language that supports all functionalities of a RDBMS i.e. data definition, data manipulation, integrity constraints, transaction management, authorization.

SQL

Codd's rules

simple view → updateable.

- Rule 6: The view updating rule:
  - All views that are theoretically updatable are also updatable by the system.

view → group by → not updatable ×

- Rule 7: Possible for high-level insert, update, and delete:
  - The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.

→ bulk DML

- Rule 8: Physical data independence:
  - Application programs and terminal activities remain logically unbroken whenever any changes are made in either storage representations or access methods.

→ not fully implemented in MySQL.
- not all query work in all engines.

→ storage engine.

- Rule 9: Logical data independence:
  - Application programs & terminal activities remain logically unbroken when information-preserving changes of any kind that theoretically permit un-impairment are made to the base tables.

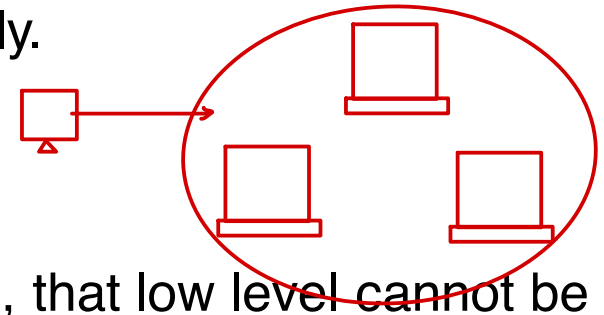
→ can be implemented using views.

Codd's rules

- Rule 10: Integrity independence:
  - Integrity constraints specific to a particular relational database must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.

① PK cannot be null.
② child row (FK) cannot be added if parent row (PK) is not avail.

- Rule 11: Distribution independence:
  - The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only.



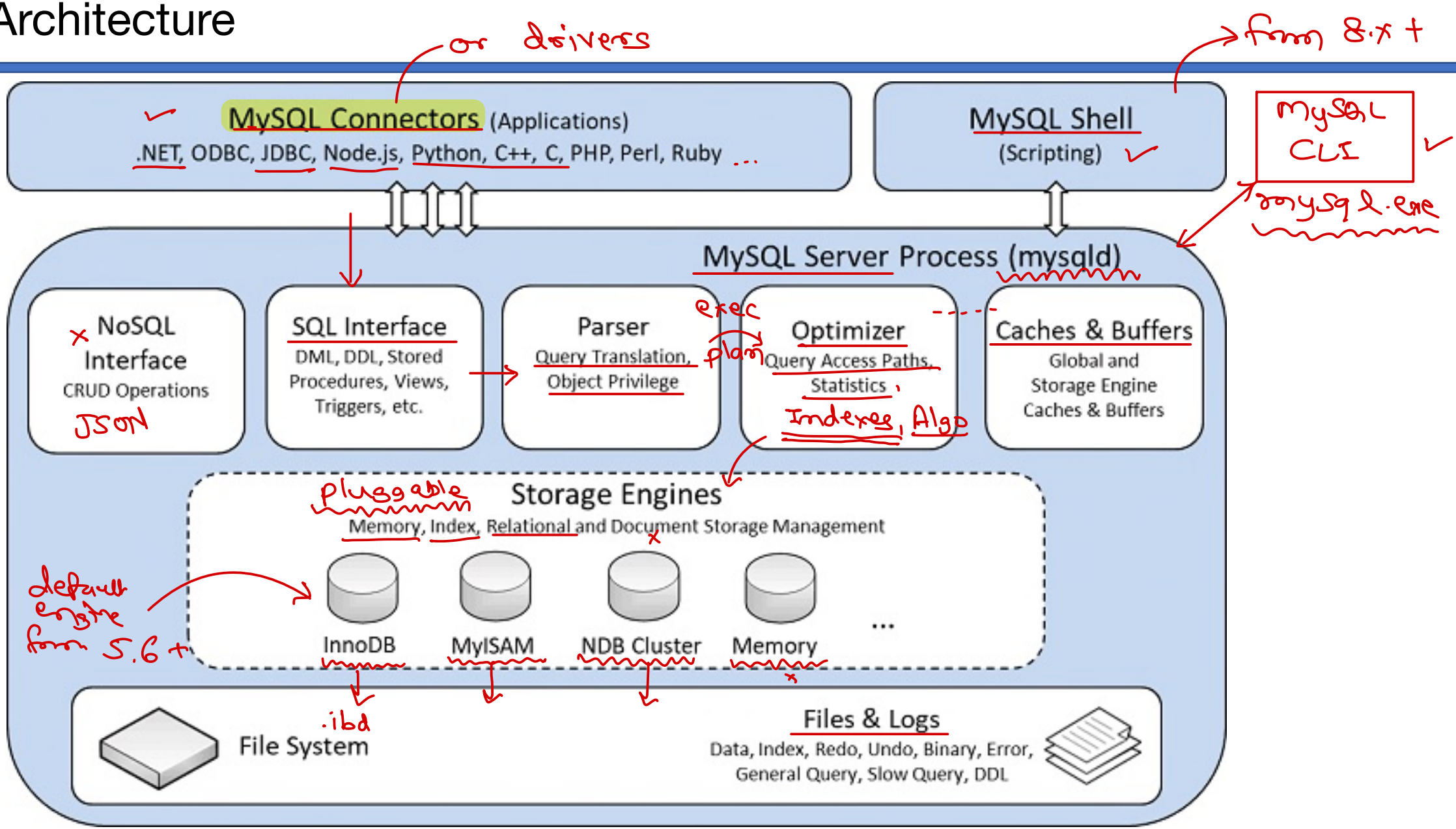
- Rule 12: The non-subversion rule:
  - If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple-records-at-a-time).

→ no backdoor entry.

Temporary Tables

* Like views, but are materialized (stored in memory).
* faster query execution.
* limited to current session.

MySQL Architecture



InnoDB vs MyISAM

if table is indexed.

- InnoDB has row-level locking. MyISAM only has full table-level locking.
- InnoDB has what is called referential integrity which involves supporting foreign keys (RDBMS) and relationship constraints, MyISAM does not (DBMS).
- InnoDB supports transactions, which means you can commit and roll back. MyISAM does not.
- InnoDB is more reliable as it uses transactional logs for auto recovery. MyISAM does not.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>