TECH VAULT

SQL QUERY LIFE CYCLE

AMR ELHELW

CRUD Operations

C	• CREATE / INSERT
R	• READ (SELECT)
U	• UPDATE
D	• DELETE

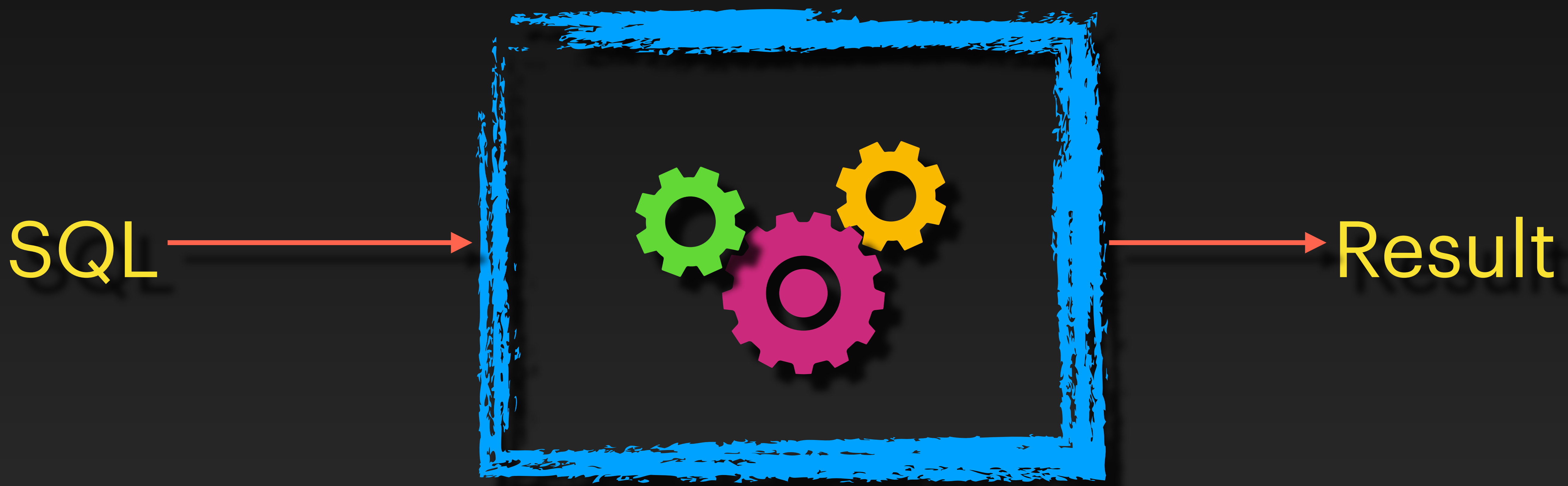
```sql
INSERT INTO employees (first_name, last_name, salary)
VALUES ('John', 'Doe', 60000);
```

```sql
SELECT first_name, last_name, salary
FROM employees
WHERE salary > 50000;
```

```sql
UPDATE employees
SET salary = salary * 1.1
WHERE salary < 50000;
```

```sql
DELETE FROM employees
WHERE employee_id = 123;
```

Query Engine



$$\begin{tikzcd}\text{SQL} \arrow[r] & \boxed{\text{Parser}}\end{tikzcd}$$

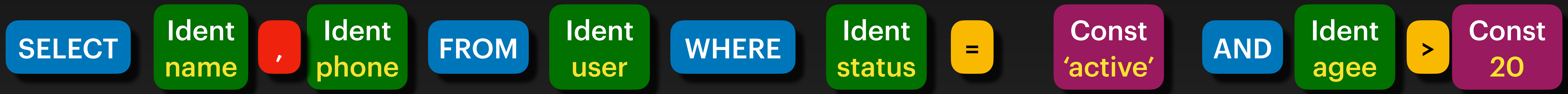
```
SELECT id, name FROM user
WHERE status = ‘A’ AND age > 20
```

is string for system and the engine converts words
so tokens stored in the system to be able to work with them

every unknown word would be considered an identifier
even if it was wrong
in this phase no checks for identifiers. it happens in analyzer

Parsing

```
SELECT name, phone FROM user WHERE status = 'active' AND agee > 20
```

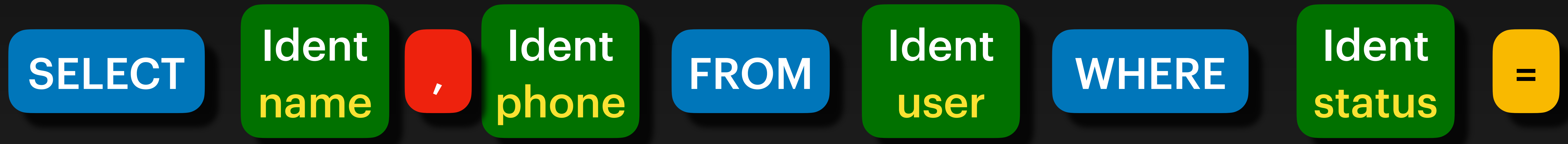


1
Tokens

Keywords	SELECT - FROM - AND - OR - WHERE - NULL - NOT - IN - GROUP -...
Operations	> - < - = - <= - >= - + - ...
Other symbols	) - (- , (comma) - . (dot) - ...
Constant values	'Tom' - 16 - '02-04-2023' - ...
Identifiers	Anything else

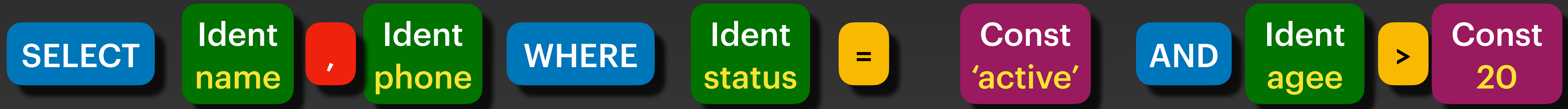
2 Syntax Checking

```
SELECT name, phone FROM user WHERE status = `active AND agee > 20
```



Missing ‘

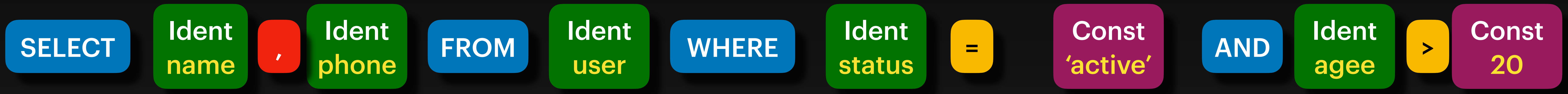
```
SELECT name, phone WHERE status = ‘active’ AND agee > 20
```

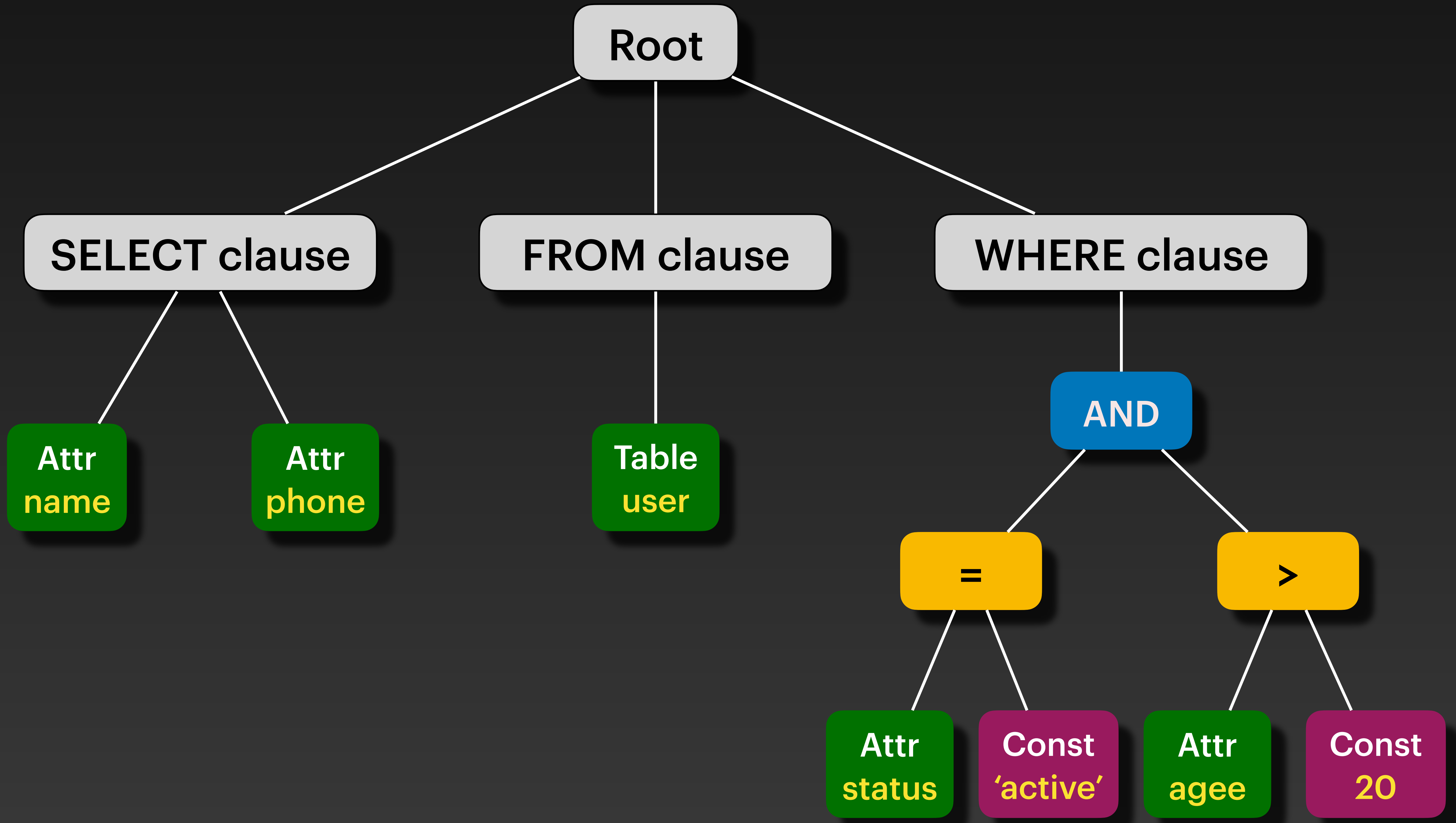


Missing FROM

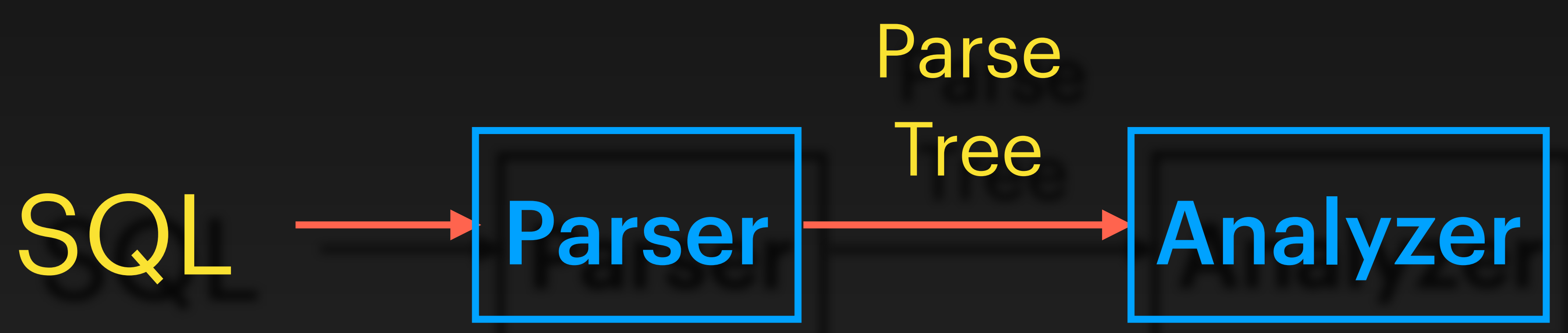
3 Parse Tree

```
SELECT name, phone FROM user WHERE status = 'active' AND agee > 20
```



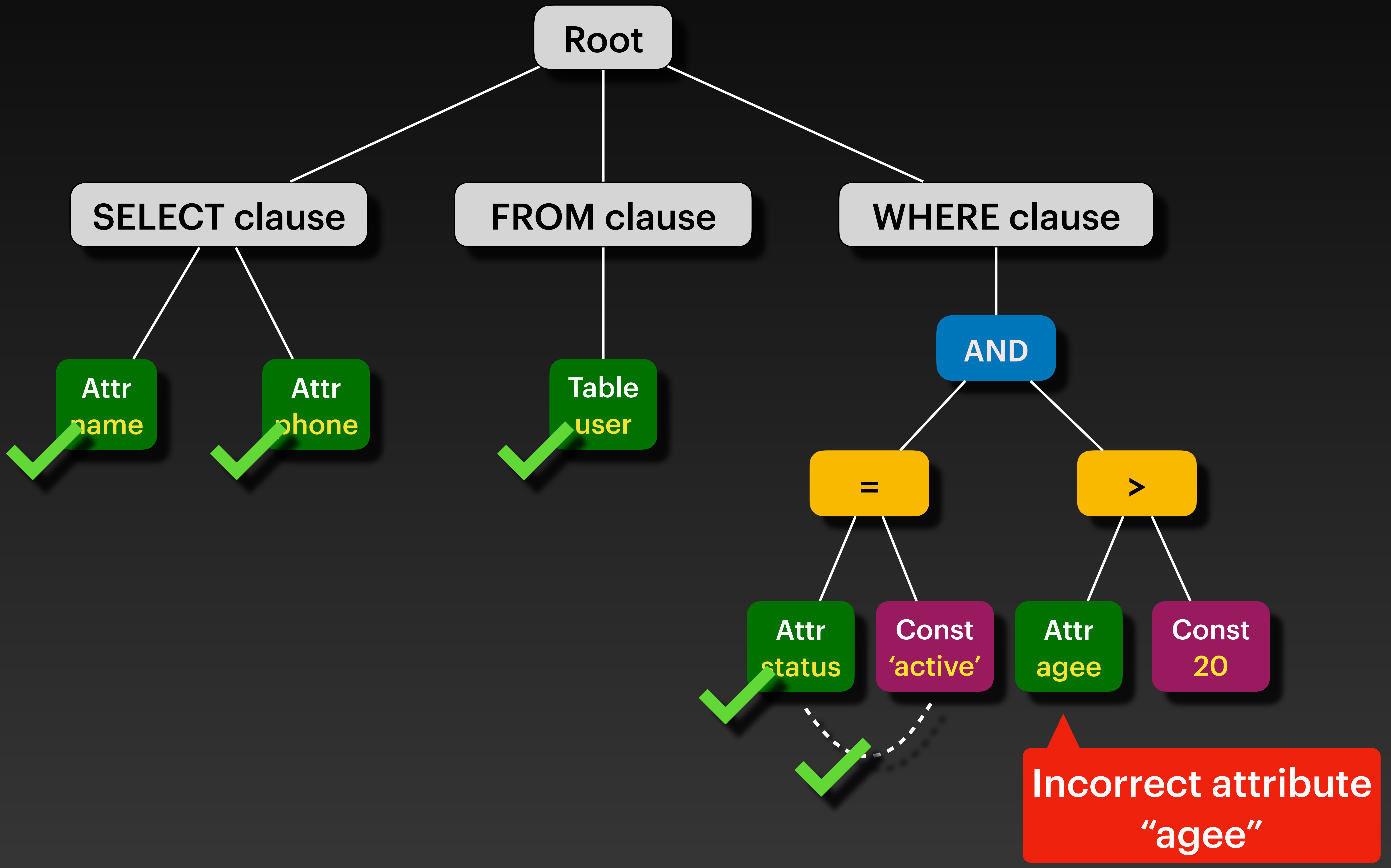


Query Engine



Query Analysis

Parse Tree

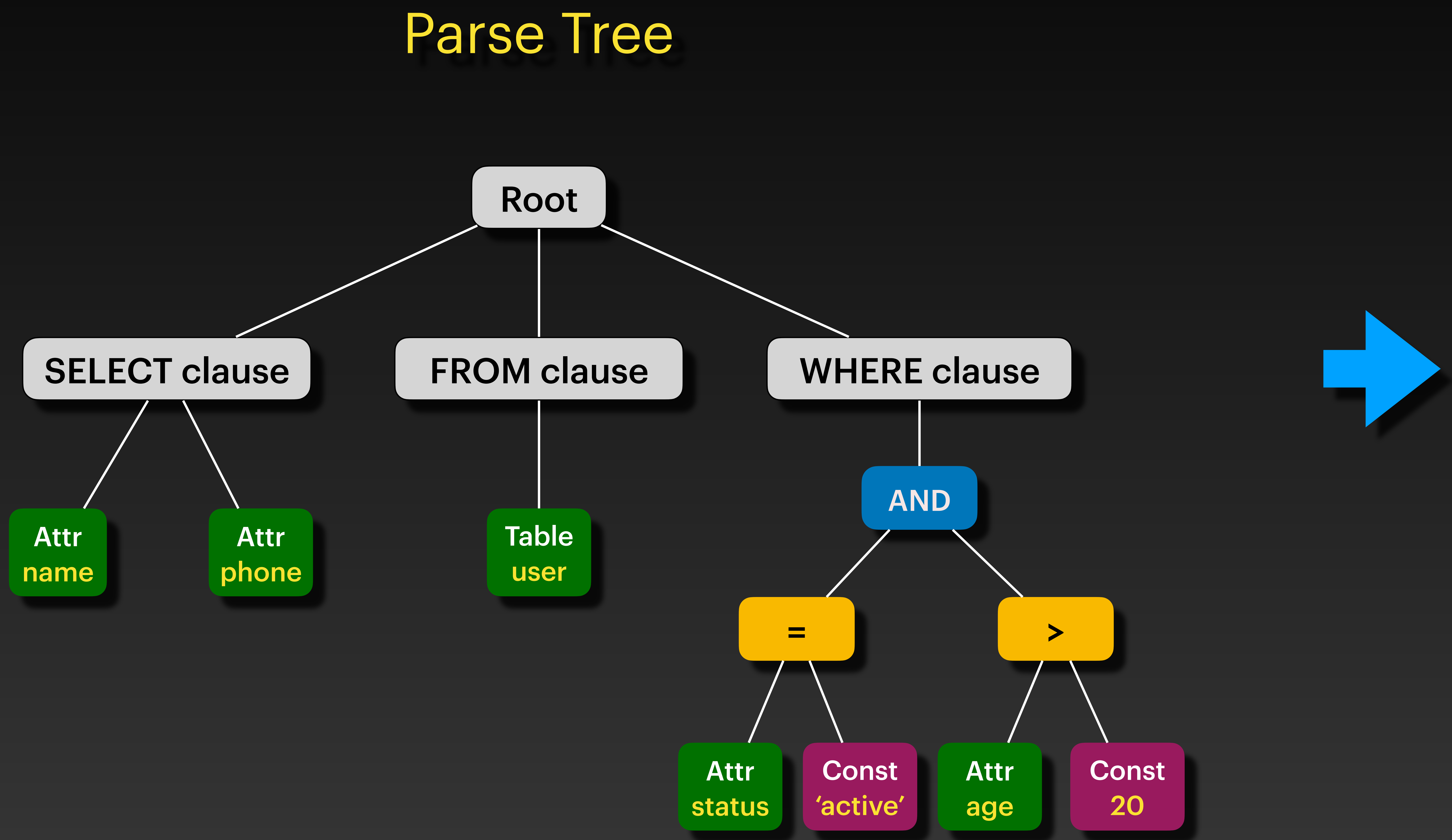


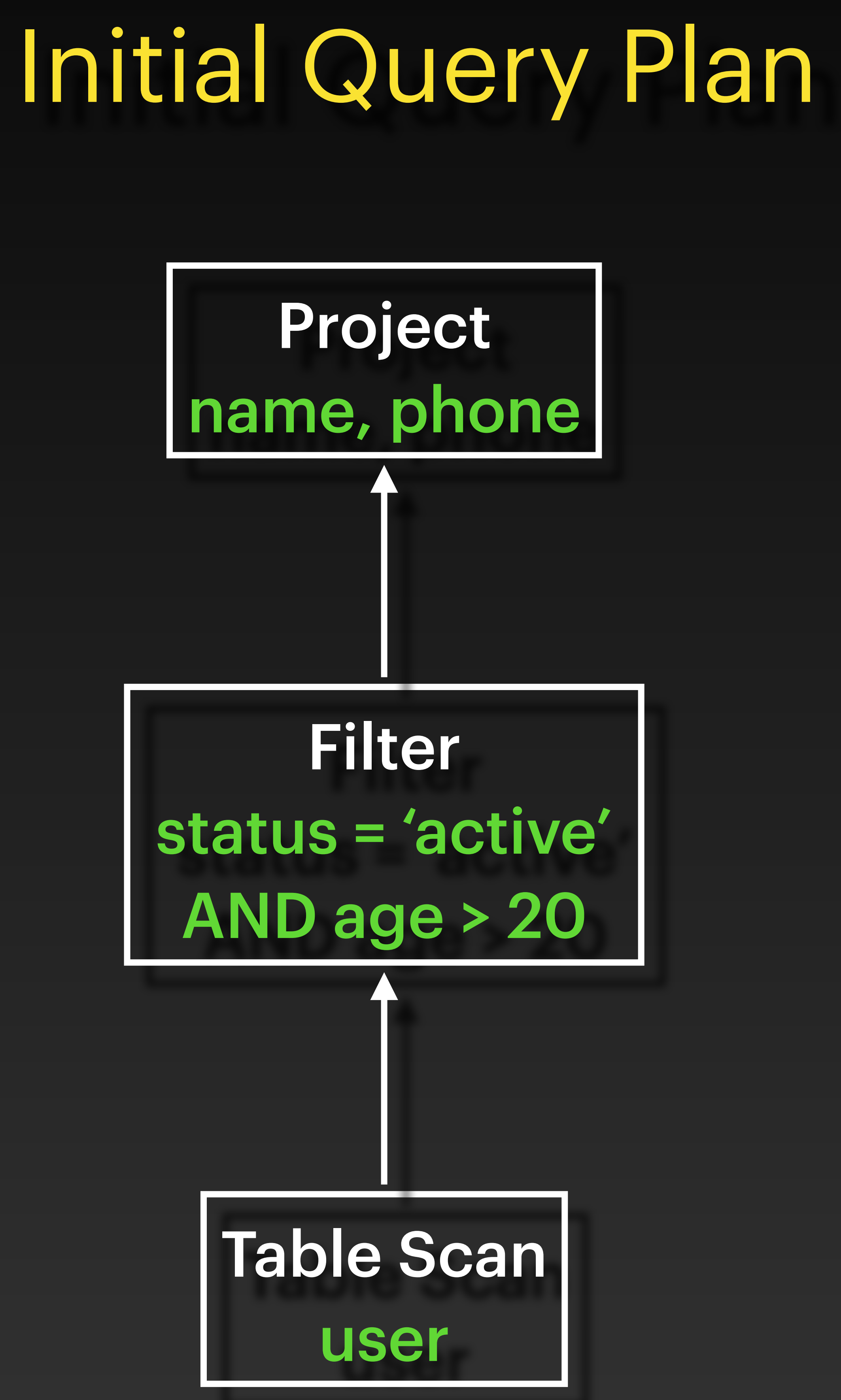
analyzer steps:
1. semantic checks:
  validates tables, cols, data types in comparison, ambigoius names, functions

Amr Elhelw's TECH VAULT

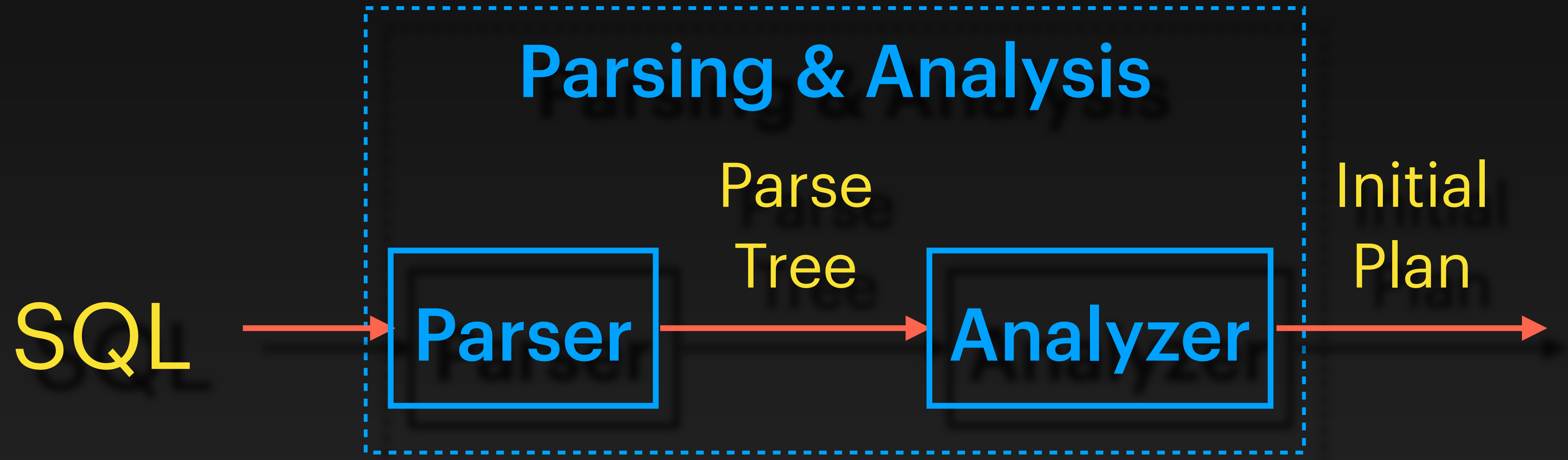
2. Initial query plan (not optimized, such as linear algebra)

Query Analysis





Query Engine



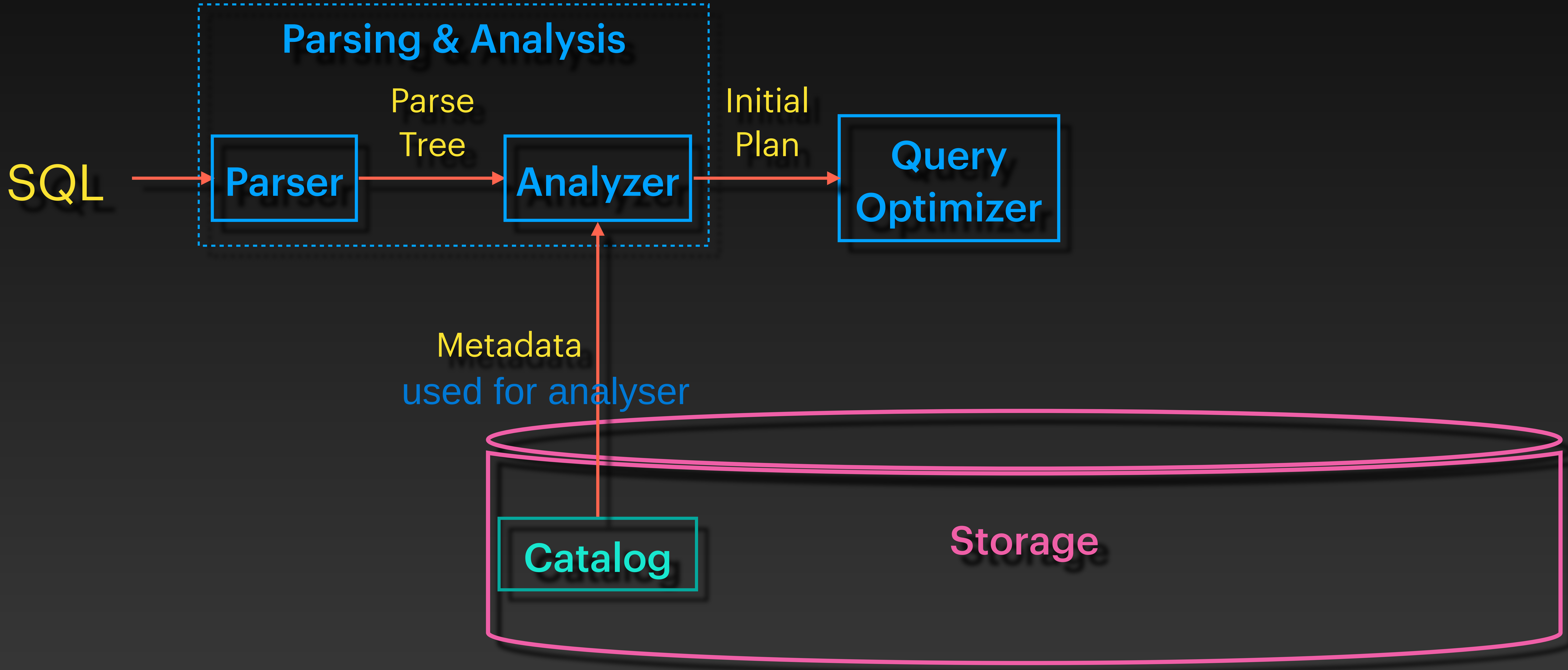
Catalog: is a namespace collection of metadata about tables, attributes, indexes and so on. they are tables also.

pg_class table: tables metadata
pg_attribute table: attributes/cols metadata
(has attrelid is forignkey from pg_class table)
pg_type table: data types metadata
pg_namespace: namespaces metadata
(such as public(pg users), catalogs in pg_catalog)

pg_stats table: statistics table contains data such as:
num of rows, distinct, nulls, avg size...etc

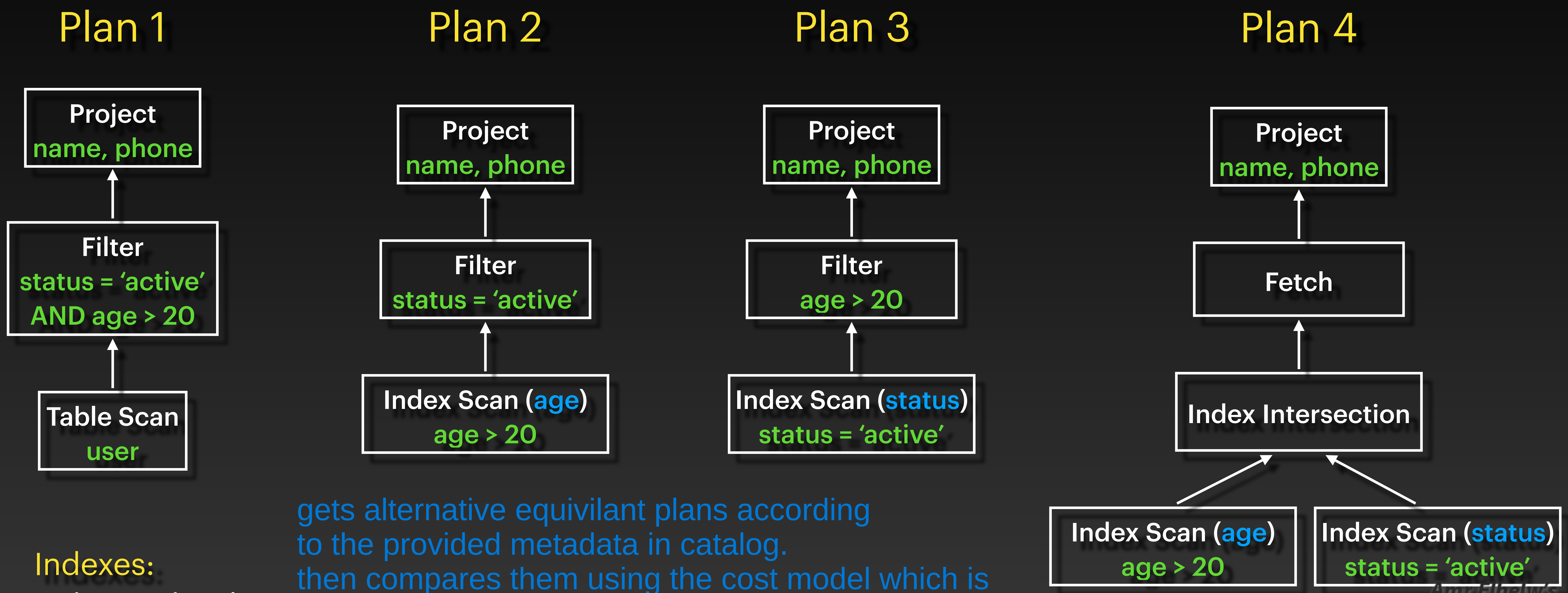
explain: gets the query plan
explain verbose: gets the returned attributes from each step in the plan
explain analyze: gets estimated and actual time and num of rows
tips:
* pgexplain website visualize the query plan from explain.
* plan are read from bottom to top
* you can format the output of explain as xml or json.

Query Engine



there is a limitless number of combinations of query plans
that no optimizer can calculates all of them which makes it not totally efficient.

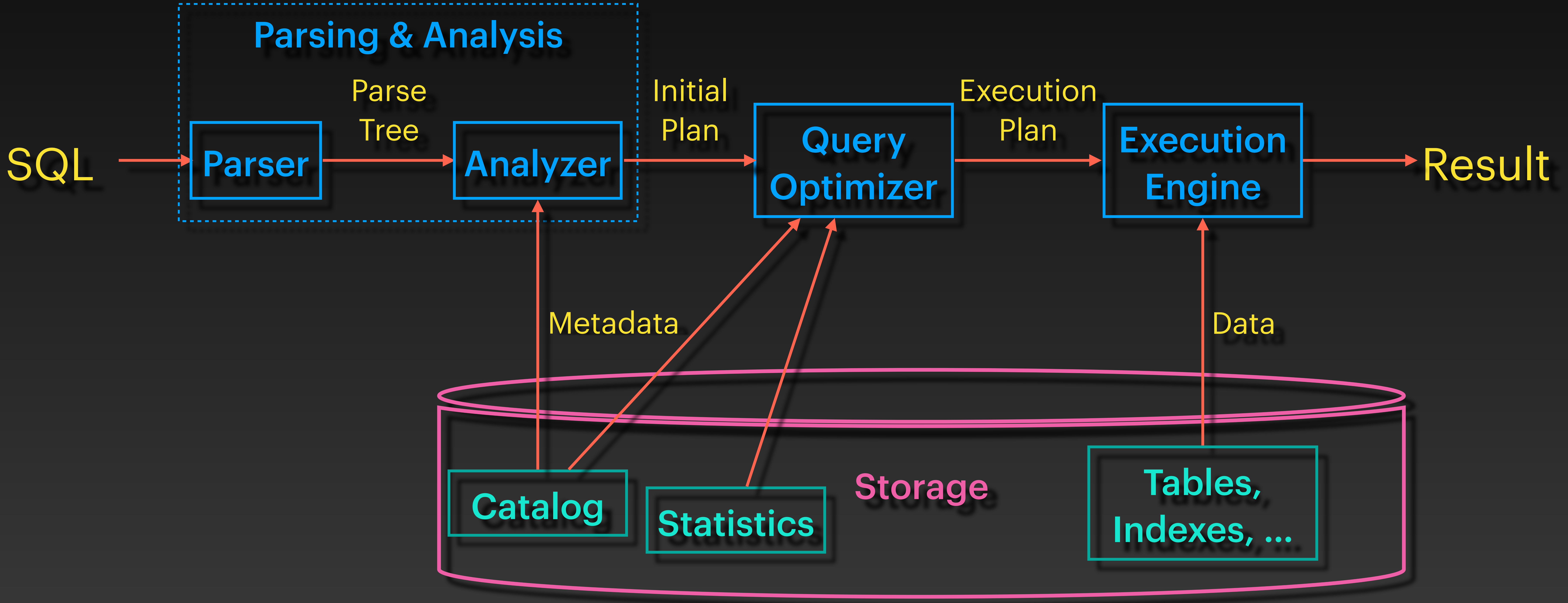
Query Optimizer



Indexes:
Index on (age)
Index on (status)

gets alternative equivilant plans according
to the provided metadata in catalog.
then compares them using the cost model which is
formulas calculations for each type of queries.
database statistics: metadata tables contains statistics about each table, number of
rows..etc

Query Engine



Parsing & Analysis
SQL
Parser
Parse Tree
Analyzer
Initial Plan
Query Optimizer
Execution Plan
Execution Engine
Result
Metadata
Data
Catalog
Statistics
Storage
Tables, Indexes, ...