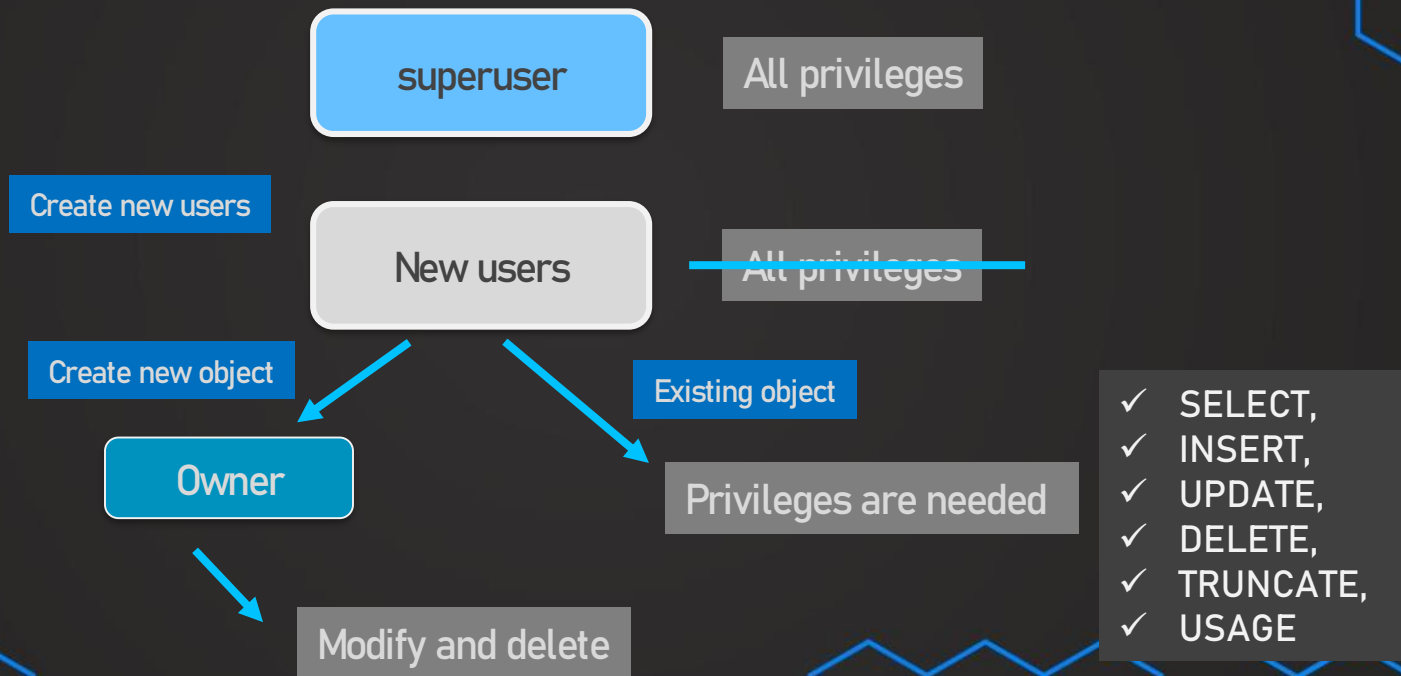


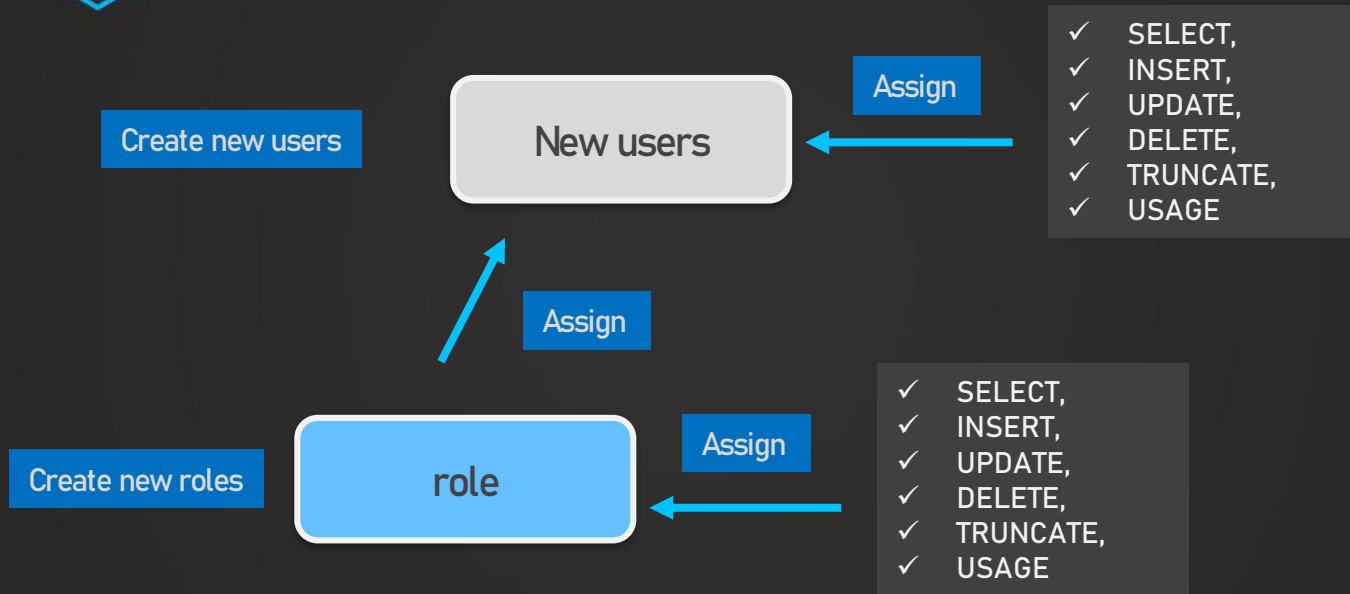
# Day 15



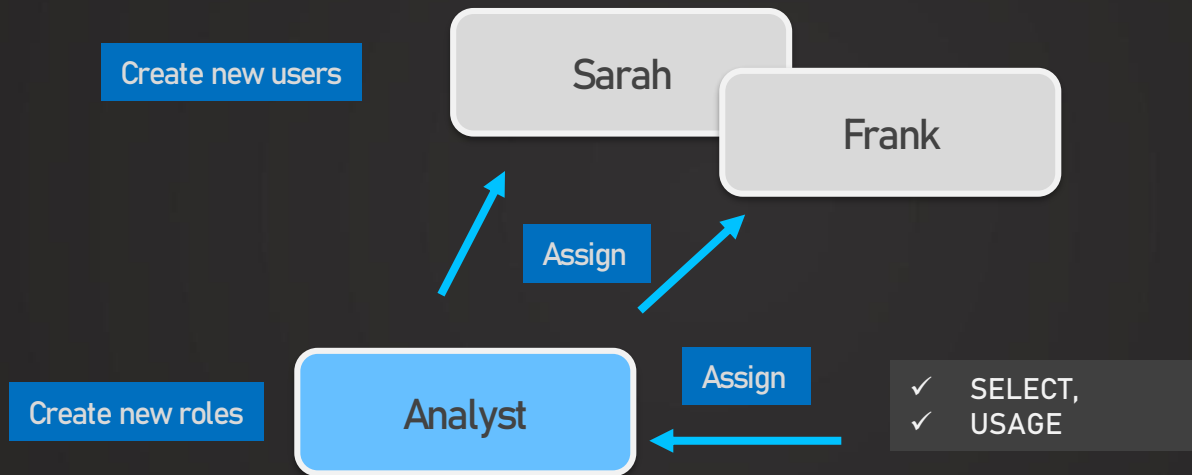
# User management



# User management

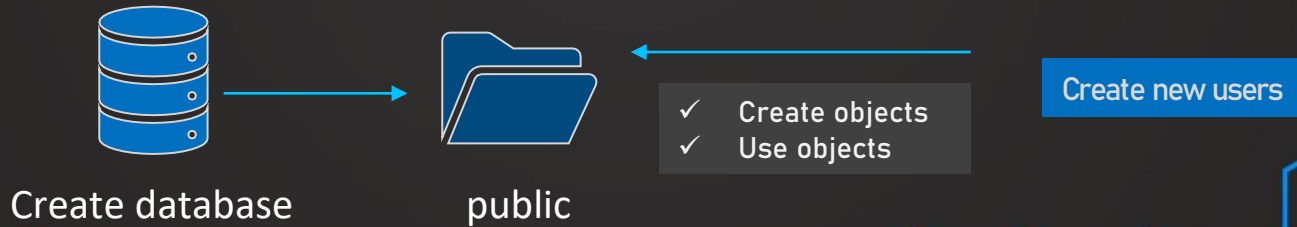


# User management



# CREATE USER

```
CREATE USER <user_name>  
WITH PASSWORD 'pwd123'
```



# CREATE USER

```
CREATE USER <user_name>  
WITH PASSWORD 'pwd123'
```

Data Output   Explain   Messages   Notifications

CREATE ROLE

Query returned successfully in 67 msec.

Role

=

User + Login

# CREATE USER

```
CREATE USER <user_name>  
WITH PASSWORD 'pwd123'
```

```
CREATE ROLE <role_name>  
WITH LOGIN PASSWORD 'pwd123'
```

Data Output Explain Messages Notifications

CREATE ROLE

Query returned successfully in 67 msec.

Role

=

User + Login

# CREATE USER

```
DROP USER <user_name>
```

```
DROP ROLE <role_name>
```



# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

Privileges

```
GRANT privilege  
ON database_object  
TO USER | ROLE | PUBLIC
```

# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

Privileges

```
GRANT SELECT  
ON customer  
TO nikolai
```

# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

TABLES

```
GRANT SELECT  
ON customer  
TO nikolai
```

# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

TABLES

```
GRANT SELECT  
ON ALL TABLES IN SCHEMA schema_name  
TO nikolai
```

# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

TABLES

```
GRANT ALL  
ON ALL TABLES IN SCHEMA schema_name  
TO nikolai
```

# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

TABLES

superuser

owner

```
GRANT ALL  
ON ALL TABLES IN SCHEMA schema_name  
TO nikolai
```

# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

TABLES

superuser

owner

```
GRANT SELECT
ON ALL TABLES IN SCHEMA schema_name
TO nikolai WITH GRANT OPTION
```

# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

Privileges

```
REVOKE privilege  
ON database_object  
FROM USER | ROLE | PUBLIC
```



# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

Privileges

```
REVOKE privilege  
ON database_object  
FROM USER | ROLE | PUBLIC  
GRANTED BY USER | ROLE
```

# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

Privileges

```
REVOKE GRANT OPTION FOR privilege
ON database_object
FROM USER | ROLE | PUBLIC
GRANTED BY USER | ROLE
```

# Privileges

Privilege	Applicable Object Types
SELECT	TABLE (and table-like objects), table column
INSERT	TABLE, table column
UPDATE	TABLE, table column
DELETE	TABLE
TRUNCATE	TABLE
CREATE	DATABASE, SCHEMA
CONNECT	DATABASE
EXECUTE	FUNCTION, PROCEDURE
USAGE	SCHEMA

# Privileges

How to grant acces?  
Typical statements

CREATE USER

```
CREATE USER amar  
WITH PASSWORD 'amar1234';
```

GRANT USAGE on schema

```
GRANT USAGE  
ON SCHEMA name  
TO amar;
```

GRANT SELECT & UPDATE

```
GRANT SELECT, UPDATE  
ON customer  
TO amar;
```

# Privileges

How to grant acces?  
Typical statements

GRANT all privileges on schema

```
GRANT ALL  
ON ALL TABLES IN SCHEMA public  
TO amar ;
```

GRANT all privileges on database

```
GRANT ALL  
ON DATABASE greencycles  
TO amar ;
```

# Privileges

How to grant acces?  
Typical statements

GRANT createdb

```
ALTER USER amar CREATEDB;
```

GRANT roles to user

```
GRANT sarah TO amar;
```

GRANT roles to user

```
GRANT analyst TO amar;
```

# Privileges

How to grant acces?  
Typical statements

REVOKE INSERT

```
REVOKE INSERT ON customer FROM amar;
```

REVOKE ALL PRIVILEGES

```
REVOKE ALL PRIVILEGES ON customer FROM PUBLIC ;
```

REVOKE ROLE

```
REVOKE analyst FROM amar;
```

# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

Privileges

```
REVOKE GRANT OPTION FOR privilege
ON database_object
FROM USER | ROLE | PUBLIC
GRANTED BY USER | ROLE
```



# Privileges

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE,
- ✓ ALL

Privileges

```
GRANT SELECT
ON ALL TABLES IN SCHEMA
<schema_name>
TO nikolai
```

# Privileges

Privileges

USAGE

Roles

Users

- ✓ SELECT,
- ✓ INSERT,
- ✓ UPDATE,
- ✓ DELETE,
- ✓ TRUNCATE,
- ✓ USAGE

```
CREATE ROLE <user_name>  
WITH LOGIN PASSWORD 'pwd123'
```

# Using indexes

**Read**  
operations

**Write**  
operations

# Using indexes

**Read**  
operations

**Write**  
operations

Understanding  
indexes

Types of  
indexes

Guidelines

Demo

# Using indexes

transaction_id [PK] integer	product_id character varying	customer_id integer	payment character varying	price numeric
1	P0494	4	visa	18.29
2	P0221	5	visa	1.49
3	P0625	5	visa	5.89
4	P0431	8	mastercard	11.59
5	P0058	5	mastercard	12.39

```
SELECT  
product_id  
FROM sales  
WHERE customer_id = 5
```

3, P0625, 5, visa

4, P0432, 8, mastercard

1, P0494, 4, visa

6, P0058, 5, mastercard

2, P0221, 5, visa

Table scan

Read-inefficient

Data is stored without a particular order

# Using indexes

transaction_id [PK] integer	product_id character varying	customer_id integer	payment character varying	price numeric
1	P0494	4	visa	18.29
2	P0221	5	visa	1.49
3	P0625	5	visa	5.89
4	P0431	8	mastercard	11.59
5	P0058	5	mastercard	12.39

```
SELECT  
  product_id  
FROM sales  
WHERE customer_id = 5
```

1, P0494, 4, visa  
6, P0058, 5, mastercard  
3, P0625, 5, visa

2, P0221, 5, visa  
4, P0432, 8, mastercard

Location	Value
1	4
2	5
5	8

## Using Index

✓ **Indexes help to make data reads faster!**

❖ **Slower data writes**

❖ **Additional storage**

❖ **B-tree Indexes**

❖ **Bitmap Indexes**

# Using indexes

transaction_id [PK] integer	product_id character varying	customer_id integer	payment character varying	price numeric
1	P0494	4	visa	18.29
2	P0221	5	visa	1.49
3	P0625	5	visa	5.89
4	P0431	8	mastercard	11.59
5	P0058	5	mastercard	12.39

```
SELECT  
  product_id  
FROM sales  
WHERE customer_id = 5
```

```
1, P0494, 4, visa  
6, P0058, 5, mastercard  
3, P0625, 5, visa
```

```
2, P0221, 5, visa  
4, P0432, 8, mastercard
```

Location	Value
1	4
2	5
5	8



# Using indexes

Location	Value
1	4
2	5
5	8

✓ Different types of indexes  
for different situations

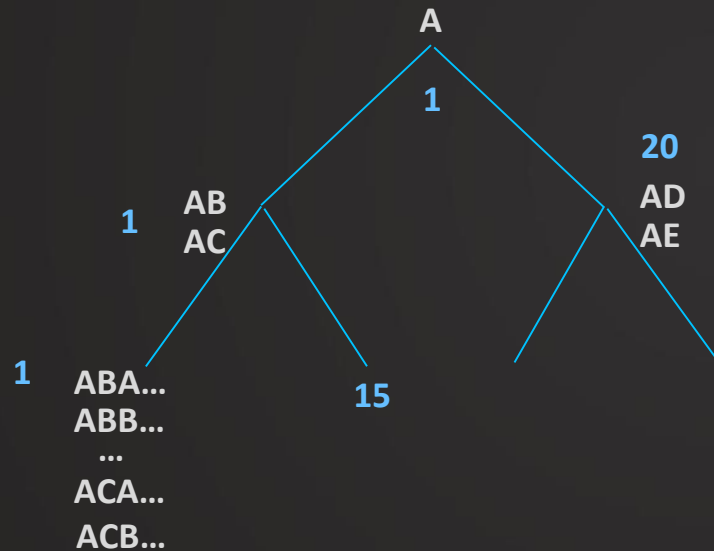
❖ B-tree Indexes

❖ Bitmap Indexes

1, P0494, 4, visa  
6, P0058, 5, mastercard  
3, P0625, 5, visa

2, P0221, 5, visa  
4, P0432, 8, mastercard

## ❖ B-tree Indexes



- ✓ Multi-level tree structure
- ✓ Breaks data down into pages or blocks
- ✓ Should be used for high-cardinality (unique) columns
- ✓ Not entire table (costly in terms of storage)



## Bitmap index

transaction_id [PK] integer	product_id character varying	customer_id integer	payment character varying	price numeric
1	P0494		4 visa	18.29
2	P0221		5 visa	1.49
3	P0625		5 visa	5.89
4	P0431		8 mastercard	11.59
5	P0058		5 mastercard	12.39

- ✓ Particularly good for dataware houses
- ✓ Large amounts of data + low-cardinality
- ✓ Very storage efficient
- ✓ More optimized for read & few DML-operations



## Bitmap index

transaction_id [PK] integer	product_id character varying	customer_id integer	payment character varying	price numeric
1	P0494		4 visa	18.29
2	P0221		5 visa	1.49
3	P0625		5 visa	5.89
4	P0431		8 mastercard	11.59
5	P0058		5 mastercard	12.39

Row_id	Value	Bit
1	visa	1 1 1 0 0
4	mastercard	0 0 0 1 1

- ✓ Particularly good for dataware houses
- ✓ Large amounts of data + low-cardinality
- ✓ Very storage efficient

Good for many repeating values  
(dimensionality)



## Bitmap index

transaction_id [PK] integer	product_id character varying	customer_id integer	payment character varying	price numeric
1	P0494		4 visa	18.29
2	P0221		5 visa	1.49
3	P0625		5 visa	5.89
4	P0431		8 mastercard	11.59
5	P0058		5 mastercard	12.39

Value	1	2	3	4	5	6	7	8
mastercard				X	X			
visa	X	X	X					

- ✓ Particularly good for dataware houses
- ✓ Large amounts of data + low-cardinality
- ✓ Very storage efficient

Good for many repeating values  
(dimensionality)

# Guidelines

## B-tree Index

Default index

Unique columns  
(surrogate key, names)

## Bitmap Index

Slow to update

Storage efficient

Great read performance

# Guidelines

Should we put index on every column?

No! They come with a cost!

Storage + Create/Update time

Only when necessary!

Avoid full table reads

Small tables do not require indexes

# Guidelines

On which columns?

1. Large tables

2. Columns that are used as filters

transaction_id [PK] integer	product_id character varying	customer_id integer	payment character varying	price numeric
1	P0494	4	visa	18.29
2	P0221	5	visa	1.49
3	P0625	5	visa	5.89
4	P0431	8	mastercard	11.59
5	P0058	5	mastercard	12.39



# Guidelines

**Fact tables**

B-tree on surrogate key

Bitmap key on foreign keys

**Dimension table**

Size of table

Are they used in searches a lot?

Choose based on cardinality

# Demo: Creating indexes

```
CREATE INDEX index_name  
ON table_name [USING method]  
(  
    column_name  
    [...]  
);
```

# Demo: Creating indexes

```
CREATE INDEX index_name  
ON table_name  
(  
    column_name  
);
```

# Demo: Creating indexes

```
CREATE INDEX index_name  
ON table_name  
(  
    column_name1,  
    column_name2  
);
```