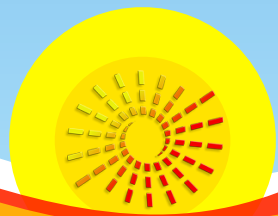


Databases and SQL syntax workshop



UPLB January 2016

Noe Fernandez-Pozo



Class Content

- Relational and not relational databases
- PostgreSQL and SQL syntax
- Table relationships and constraints

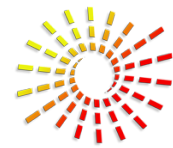


Relational Databases

Databases structured to recognize relations among stored items of information.







SGN Database



genes, genomes, sequences, maps, markers, users, expression data, phenotypic data, breeding data, blast management ...

<https://solgenomics.net>

+

CHADO

An ontology-based modular schema
for representing genome-associated
biological information



Non Relational Databases

Non-relational databases, also called NoSQL databases.



- GBS
- Breeding data
- Phenotypic data
- Variation data



Indexed Files

- BLAST databases
- Bowtie indexes
- JBrowse



SQL Language



The world's most advanced
open source database

<http://www.postgresql.org>

<http://www.postgresql.org/docs/9.4/static/sql-commands.html>



Connect to a database for the first time

install postgresSQL

```
sudo aptitude install postgresql
```

```
sudo -u postgres psql postgres
```

connect for the first time

UNIX
Terminal

```
\password user
```

change user password

```
\du
```

list roles

postgresql
Terminal



Connect to a PostgreSQL Database

psql help

```
psql --help
```

```
man psql
```

psql manual



Connect to a PostgreSQL database

Connect to the database

Connect to the database using postgres user

```
psql -h host -U user
```

```
psql -h localhost -U postgres
```

```
psql -h localhost -U bioinfo -d pineapple_backup
```

Connect to the database using bioinfo user



Create and alter a role

create role with permission to create databases

\du

list roles

```
CREATE ROLE bioinfo WITH LOGIN ENCRYPTED PASSWORD 'bioinfo' CREATEDB;
```

```
ALTER ROLE bioinfo CREATEROLE CREATEDB REPLICATION SUPERUSER;
```

```
DROP ROLE bioinfo;
```

delete role bioinfo

grant all permissions to bioinfo user

<http://www.postgresql.org/docs/9.4/static/sql-alterrole.html>



SQL syntax

- Create, edit and delete databases and schemas
- Import and export databases
- Create, edit and delete tables
- Import tables from a file
- Transactions: begin, rollback and commit
- Insert data and update data in databases
- Explore databases, schemas and tables
- Select and conditions: limit, offset, distinct, where, etc.
- Basic table joining



Basic Commands



PostgreSQL Cheat Sheet

BTI-SGN Databases Course 2016



List database, schemas and tables		SQL query commands
\?	show help	SELECT * FROM mytable;
\h	SQL syntax help	SELECT column1 FROM mytable;
\h create table	SQL help for create table	SELECT ... WHERE mycolumn = 'value1';
\l	list databases	SELECT DISTINCT mycolumn from mytable;
\c mydb	connect to mydb database	SELECT ... GROUP BY mycolumn;
\dn	list schemas from a database	SELECT ... WHERE mycolumn IN (1,2);
\dt	list tables from public schema	SELECT ... WHERE mycolumn LIKE '%abc%';
\dt sgn.*	list tables from schema sgn	SELECT ... WHERE mycolumn ILIKE '%abc%';
\d blast_db	show blast_db table details	SELECT ... WHERE mycolumn IS NULL;
\q	quit psql	SELECT ... LIMIT 10;
		SELECT ... OFFSET 10;
		SELECT ... JOIN mytable using(common_column);
		SELECT ... JOIN mytable on(common_column1=common_column2)

SQL development commands
CREATE DATABASE mydb;
ALTER DATABASE mydb RENAME TO newname;
DROP DATABASE mydb;
CREATE TABLE mytable;
ALTER TABLE mytable RENAME TO newname;
DROP TABLE mytable CASCADE;
ALTER TABLE mytable ADD COLUMN column_name column_type;
ALTER TABLE mytable RENAME COLUMN column_name TO newname;
ALTER TABLE mytable DROP COLUMN column_name RESTRICT;
INSERT INTO mytable (column1,column2) VALUES ('value1','value2');
UPDATE mytable SET column1='value1' WHERE column=value;
DELETE FROM mytable WHERE column='value';
begin;
commit;
rollback;



Basic Commands

\?

show help

\h

SQL syntax help

<http://www.postgresql.org/docs/9.4/static/sql-commands.html>

\h create database

SQL syntax help

\l

List databases

\c mydb

Connect to a database

\q

quit psql



Basic Commands

`\dn` — list schemas from the database

`\dt` — list all tables from public schema

`\dt sgn.*` — list all tables from sgn schema

`\d blast_db` — list all tables from sgn schema

`\o file.txt` — send output to file



SQL Development Commands



PostgreSQL Cheat Sheet

BTI-SGN Databases Course 2016



List database, schemas and tables		SQL query commands
\?	show help	SELECT * FROM mytable;
\h	SQL syntax help	SELECT column1 FROM mytable;
\h create table	SQL help for create table	SELECT ... WHERE mycolumn = 'value1';
\l	list databases	SELECT DISTINCT mycolumn from mytable;
\c mydb	connect to mydb database	SELECT ... GROUP BY mycolumn;
\dn	list schemas from a database	SELECT ... WHERE mycolumn IN (1,2);
\dt	list tables from public schema	SELECT ... WHERE mycolumn LIKE '%abc%';
\dt sgn.*	list tables from schema sgn	SELECT ... WHERE mycolumn ILIKE '%abc%';
\d blast_db	show blast_db table details	SELECT ... WHERE mycolumn IS NULL;
\q	quit psql	SELECT ... LIMIT 10;
		SELECT ... OFFSET 10;
		SELECT ... JOIN mytable using(common_column);
		SELECT ... JOIN mytable on(common_column1=common_column2)

SQL development commands
CREATE DATABASE mydb;
ALTER DATABASE mydb RENAME TO newname;
DROP DATABASE mydb;
CREATE TABLE mytable;
ALTER TABLE mytable RENAME TO newname;
DROP TABLE mytable CASCADE;
ALTER TABLE mytable ADD COLUMN column_name column_type;
ALTER TABLE mytable RENAME COLUMN column_name TO newname;
ALTER TABLE mytable DROP COLUMN column_name RESTRICT;
INSERT INTO mytable (column1,column2) VALUES ('value1','value2');
UPDATE mytable SET column1='value1' WHERE column=value;
DELETE FROM mytable WHERE column='value';
begin;
commit;
rollback;



Create Database

create a database

```
CREATE DATABASE db_name;
```

create a database and define owner and encoding

```
CREATE DATABASE db_name WITH OWNER user1 ENCODING 'UTF8';
```

```
\1
```

list all databases

```
CREATE DATABASE pineapple_db;
```

create a database called pineapple_db



Grant Permissions

grant permissions to an user on a database

```
GRANT ALL PRIVILEGES ON DATABASE db_name TO user1;
```

```
SELECT ("read")  
UPDATE ("write")  
INSERT ("append")  
DELETE  
TRUNCATE  
REFERENCES  
CREATE  
CONNECT  
TEMPORARY  
ALL PRIVILEGES
```



Grant Permissions

```
\dp mytable
```

			Access privileges	
Schema	Name	Type	Access privileges	Column access privileges
public	mytable	table	bioinfo=arwdDxt/bioinfo : =r/bioinfo : admin=arw/bioinfo	col1: : bioinfo_rw=rw/bioinfo

```
rolename=xxxx -- privileges granted to a role  
=xxxx -- privileges granted to PUBLIC
```

```
  r -- SELECT ("read")  
  w -- UPDATE ("write")  
  a -- INSERT ("append")  
  d -- DELETE  
  D -- TRUNCATE  
  x -- REFERENCES  
  t -- TRIGGER  
  X -- EXECUTE  
  U -- USAGE  
  C -- CREATE  
  c -- CONNECT  
  T -- TEMPORARY  
arwdDxt -- ALL PRIVILEGES (for tables, varies for other objects)  
  * -- grant option for preceding privilege  
  
/yyyy -- role that granted this privilege
```



Edit a Database

rename database db_name to new_db_name

```
ALTER DATABASE db_name RENAME TO new_db_name;
```

```
ALTER DATABASE db_name OWNER TO new_owner;
```

change database owner



How To Delete a Database

delete a database

```
DROP DATABASE db_name;
```

```
DROP DATABASE pineapple_db;
```

delete a database called pineapple_db



Dump a database

back up a database

ask for password

```
pg_dump -U user -h host -W db_name > db_file
```

```
pg_dump -U postgres -h localhost pineapple_db > pineapple_dump.sql
```

back up a database called pineapple_db



Import a Dumped Database

```
CREATE DATABASE pineapple_db;
```

create a database

import a database

```
cat pineapple_dump.sql | psql -U postgres -h localhost -d pineapple_db
```



Exercises I

1. Print in a file the table info (not the data) for all the tables in pineapple_backup
2. Create and delete a user with your name (no bioinfo)
3. Create a database called new_pineapple
4. Rename that database to pineapple_demo
5. Export the database pineapple_backup in a file called pineapple_backup.sql
6. Import the database pineapple_backup into pineapple_demo



Table Estructure

Columns

primary key column	file_base column	title column	type column
blast_db_id	file_base	title	type
1	unigene/Capsicum_combined	Capsicum annuum (pepper) Unigenes	nucleotide
2	unigene Ipomoea_batatas	Ipomoea batatas (sweet potato) Unigenes	nucleotide
3	unigene/Nicotiana_sylvestris	Nicotiana sylvestris (wood tobacco) Unigenes	nucleotide
4	unigene/Petunia_hybrida	Petunia hybrida Unigenes	nucleotide
5	genbank/nt	NCBI NT comprehensive nucleotide database	nucleotide

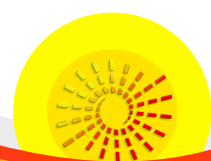


Table Estructure

Row

blast_db_id	file_base	title	type
1	unigene/Capsicum_combined	Capsicum annuum (pepper) Unigenes	nucleotide
2	unigene Ipomoea_batatas	Ipomoea batatas (sweet potato) Unigenes	nucleotide
3	unigene/Nicotiana_sylvestris	Nicotiana sylvestris (wood tobacco) Unigenes	nucleotide
4	unigene/Petunia_hybrida	Petunia hybrida Unigenes	nucleotide
5	genbank/nt	NCBI NT comprehensive nucleotide database	nucleotide



Constraints

blast_db_id	file_base	title	type
31	unigene/Capsicum_combined	Capsicum annuum (pepper) protein sequences	protein
71	unigene/Ipomoea_batatas	Ipomoea batatas (sweet potato) Unigenes	nucleotide
70	unigene/Nicotiana_sylvestris	Nicotiana sylvestris (wood tobacco) Unigenes	nucleotide
29	unigene/Petunia_hybrida	Petunia hybrida Unigenes	nucleotide
15	genbank/nt	NCBI NT comprehensive nucleotide database	nucleotide

UNIQUE NOT NULL

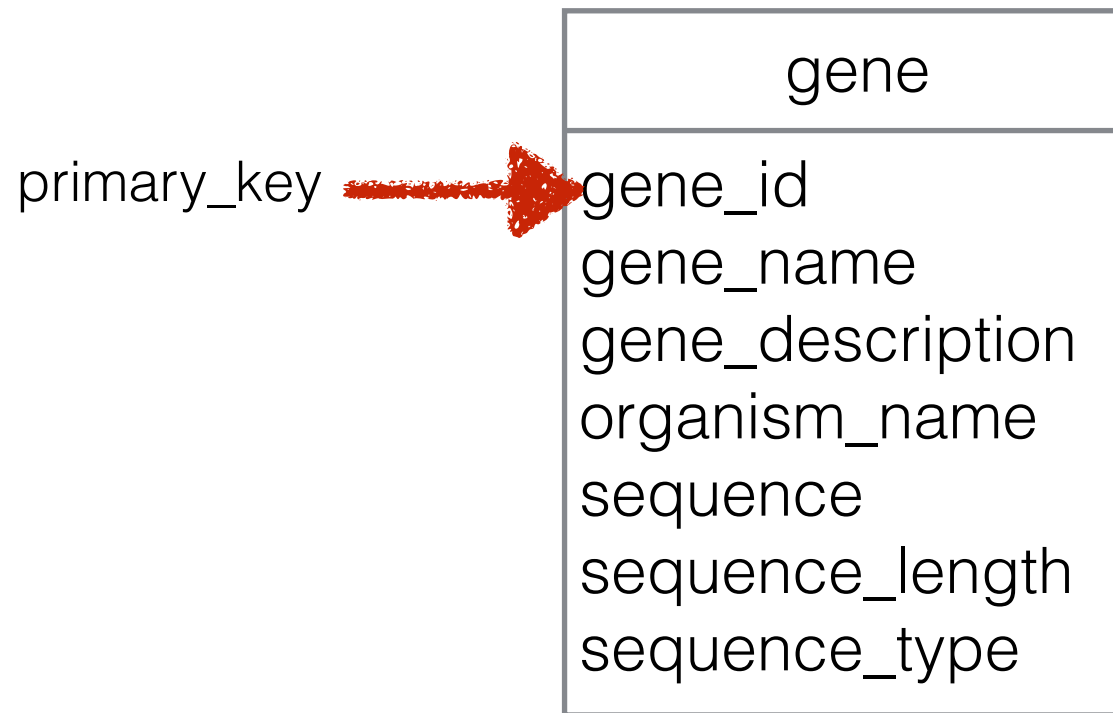
nucleotide
or
protein

- **NOT NULL Constraint:** Ensures that a column cannot have NULL value.
- **UNIQUE Constraint:** Ensures that all values in a column are different.
- **PRIMARY Key:** Uniquely identifies each row/record in a database table.
- **FOREIGN Key:** Constrains data based on columns in other tables.
- **CHECK Constraint:** The CHECK constraint ensures that all values in a column satisfy certain conditions.



Database Schema

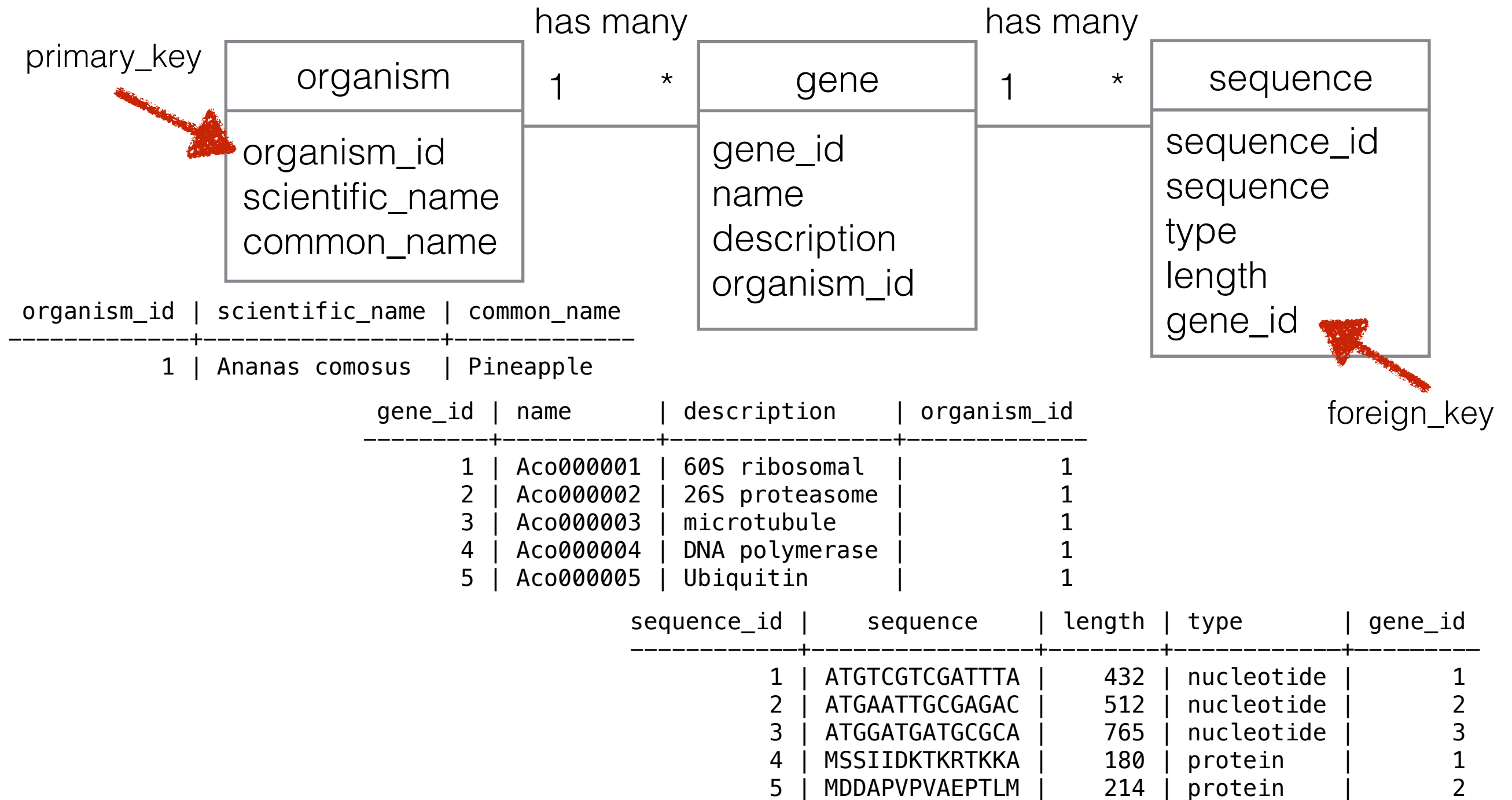
Relationships and Normalization



gene_id	gene_name	gene_description	organism_name	sequence	sequence_length	sequence_type
1	Aco000001	60S ribosomal	Pineapple	ATGTCGTCGATTTA	432	nucleotide
2	Aco000002	26S proteasome	Pineapple	ATGAATTGCGAGAC	512	nucleotide
3	Aco000003	microtubule	Pineapple	ATGGATGATGCGCA	765	nucleotide
4	Aco000004	DNA polymerase	Pineapple	MSSIIDKTKRTKKA	180	protein
5	Aco000005	Ubiquitin	Pineapple	MDDAPVPVAEPTLM	214	protein
6	Aco000001	60S ribosomal	Pineapple	MSSIIDKTKRTKKA	142	protein
7	Aco000002	26S proteasome	Pineapple	MDDAPVPVAEPTLM	174	protein



One To Many



One To Many

gene_id	gene_name	gene_description	organism_name	sequence	sequence_length	sequence_type
1	Aco000001	60S ribosomal	Pineapple	ATGTCGTCGATTTA	432	nucleotide
2	Aco000002	26S proteasome	Pineapple	ATGAATTGCGAGAC	512	nucleotide
3	Aco000003	microtubule	Pineapple	ATGGATGATGCGCA	765	nucleotide
4	Aco000004	DNA polymerase	Pineapple	MSSIIDKTKRTKKA	180	protein
5	Aco000005	Ubiquitin	Pineapple	MDDAPVPVAEPTLM	214	protein

gene_id	name	description	organism_id
1	Aco000001	60S ribosomal	1
2	Aco000002	26S proteasome	1
3	Aco000003	microtubule	1
4	Aco000004	DNA polymerase	1
5	Aco000005	Ubiquitin	1

organism_id | scientific_name | common_name

1	Ananas comosus	Pineapple
---	----------------	-----------

sequence_id	sequence	length	type	gene_id
1	ATGTCGTCGATTTA	432	nucleotide	1
2	ATGAATTGCGAGAC	512	nucleotide	2
3	ATGGATGATGCGCA	765	nucleotide	3
4	MSSIIDKTKRTKKA	180	protein	1
5	MDDAPVPVAEPTLM	214	protein	2



Many To Many

blast_db
blast_db_id
name
description
path
group
group_order

BLAST databases
classified in categories

BLAST

⊖ Input parameters

Categories Popular datasets     

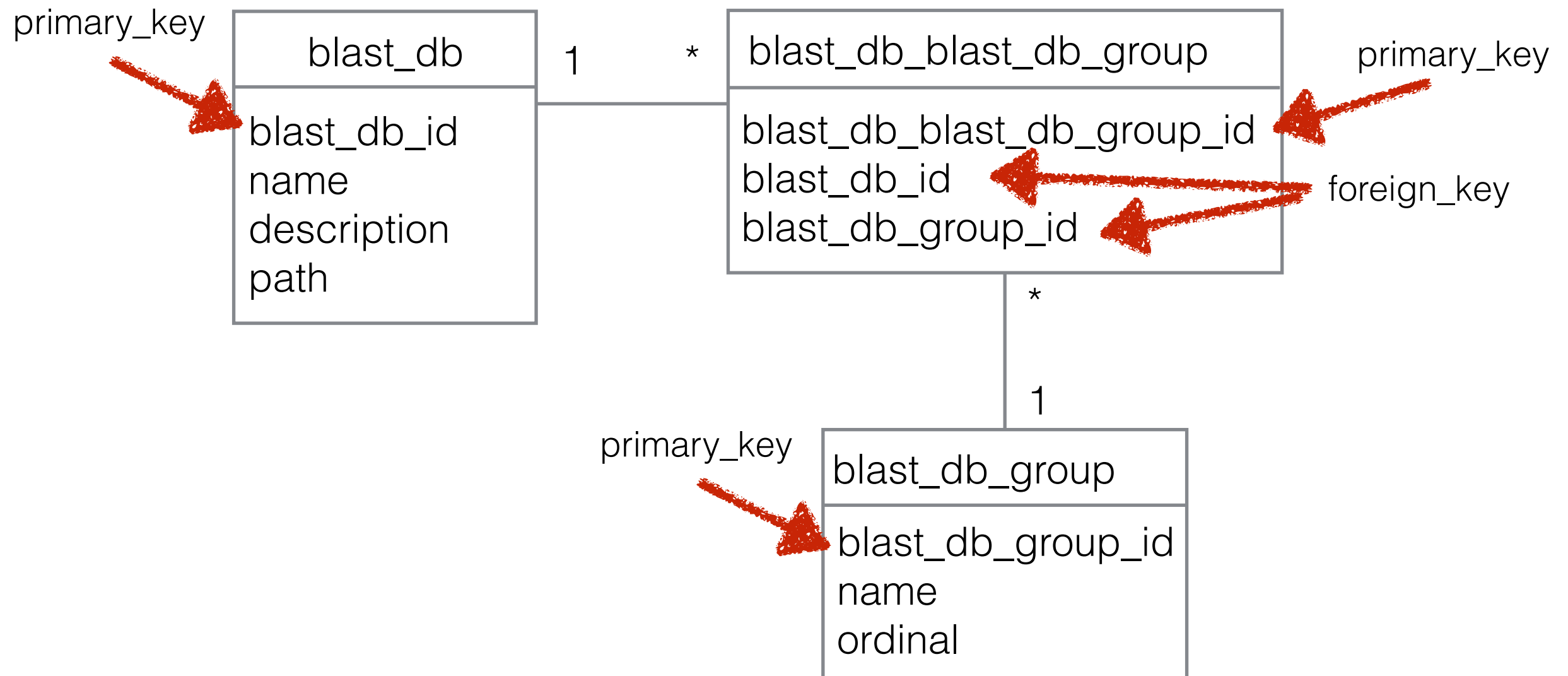
Database Tomato Genome cDNA (ITAG release 2.40) [db details](#)

✓ Popular datasets

- Tomato Genome (Current version)
- Potato Genome (Current version)
- Pepper Genome (Current version)
- Eggplant Genome (Current version)
- N.benthamiana Genome (Current version)
- N.tabacum Genomes (Current version)
- Other Nicotiana Genomes (Current version)
- Tomato Wild Species
- Coffee Genome
- Tomato Inbred Lines
- Genome Sequences
- Markers
- Organelle Genomes
- Tomato Genome (other datasets)



Many To Many Relationship



Create a Schema

create a schema

```
CREATE SCHEMA schema_name;
```

```
CREATE SCHEMA IF NOT EXISTS schema_name AUTHORIZATION user_name;
```

create a schema *schema_name* owned by *user_name* if it does not exist already



Edit a Schema

rename schema name to new_name

```
ALTER SCHEMA name RENAME TO new_name;
```

```
ALTER SCHEMA name OWNER TO new_owner;
```

change schema owner



How To Delete a Schema

delete a schema

```
DROP SCHEMA schema_name CASCADE;
```

```
DROP SCHEMA blast RESTRICT;
```

delete a schema called blast

Refuse to drop the schema if it contains any objects. This is the default.



Create a New Table

create an empty table

```
CREATE TABLE table_name ();
```

```
CREATE TABLE sgn.blast_db ();
```

create an empty table blast_db in sgn schema



Create a Table

large autoincrementing integer

row unique id

this column cannot be empty

```
CREATE TABLE gene (  
  gene_id bigserial PRIMARY KEY,  
  name varchar(80) UNIQUE NOT NULL,  
  description text,  
  length integer,  
  sequence text,  
  organism_id bigserial REFERENCES organism (organism_id)  
);
```

the value is a number

link to another table row id

gene
gene_id
name
description
length
sequence
organism_id



Create a Table

value limited to 80 variable characters

values cannot be repeated

```
CREATE TABLE gene (  
  gene_id bigserial PRIMARY KEY,  
  name varchar(80) UNIQUE NOT NULL,  
  description text,  
  length integer,  
  sequence text,  
  organism_id bigserial REFERENCES organism (organism_id)  
);
```

variable characters unlimited length

gene
gene_id
name
description
length
sequence
organism_id



Numeric Types

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807



Edit Tables

add a column of type varchar to a table

rename a column

```
ALTER TABLE mytable ADD COLUMN mycolumn varchar(30);
```

```
ALTER TABLE gene RENAME COLUMN name TO gene_name;
```

```
ALTER TABLE gene ALTER COLUMN name TYPE varchar(40);
```

```
ALTER TABLE gene DROP COLUMN sequence RESTRICT;
```

drop a column from a table

change column type



Delete a Table Using Drop

delete a table

```
DROP TABLE table_name CASCADE;
```

```
DROP TABLE gene RESTRICT;
```

delete a table called gene

Refuse to drop the table if any objects depend on it. This is the default.



Delete all rows from a table

delete rows from a table

empty all the rows from tables gene and blast_db

```
TRUNCATE table_name CASCADE;
```

```
TRUNCATE gene, blast_db;
```

```
TRUNCATE gene RESTART IDENTITY;
```

empty the table gene and restart primary id index



Delete a Table

empty a table

```
DELETE FROM table_name;
```

```
DELETE FROM gene WHERE column='value';
```

delete the rows from table gene based on a condition



Drop Vs. Truncate Vs Delete



DROP TABLE -- remove a table

DROP TABLE removes tables from the database. Only the table owner, the schema owner, and superuser can drop a table. To empty a table of rows without destroying the table, use [DELETE](#) or [TRUNCATE](#).

To drop a table that is referenced a foreign-key constraint of another table, CASCADE must be specified. (CASCADE will only remove the foreign-key constraint, not the other table entirely.)

DELETE -- delete rows of a table based on a condition

DELETE deletes rows that satisfy the WHERE clause from the specified table. If the WHERE clause is absent, the effect is to delete all rows in the table. The result is a valid, but empty table.

TRUNCATE -- empty a table or set of tables

TRUNCATE quickly removes all rows from a set of tables. Very useful on large tables.



Create New Tables From SQL File

create tables from a file

ask for password

print input from file

```
psql -U username -d myDataBase -W -a -f myInsertFile
```

```
psql -U postgres -d ananas -h localhost -a -f gene_tables.sql
```

create tables from gene_tables.sql file in pineapple_demo database



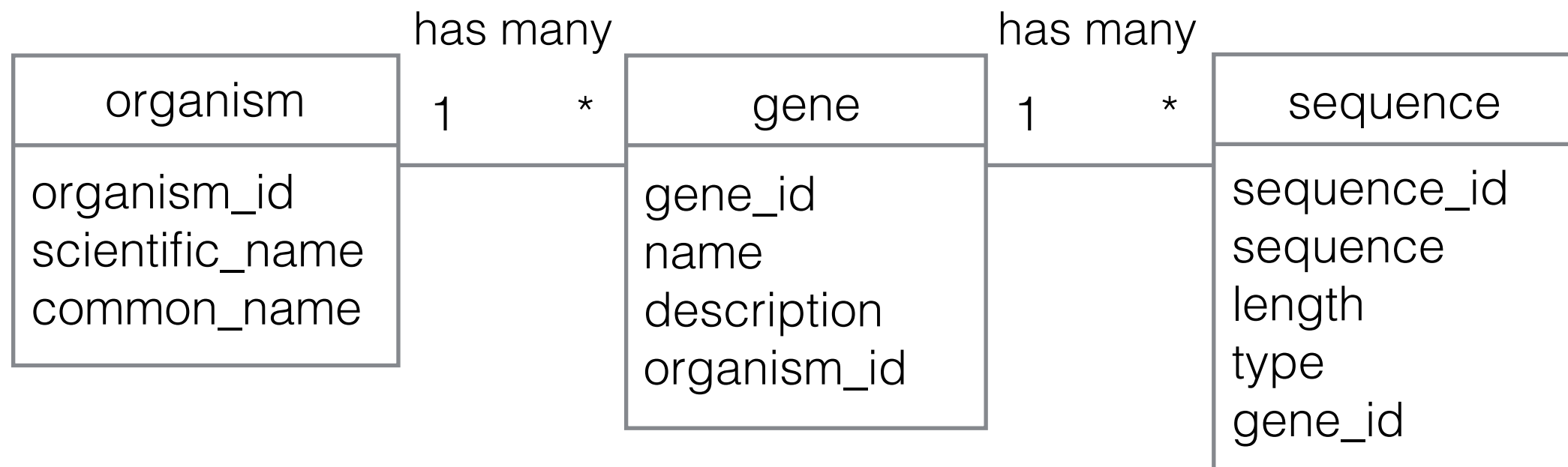
Exercises II

1. Create a database called ananas
2. Create a table called “Genes” with the columns:
 - name
 - description
 - sequence
3. Rename the table “Genes” to “gene”
4. Remove the column “sequence”
5. Add a column named organism_id
6. What are the “gene” table column types? (integer, text...)
7. Drop the table “gene”

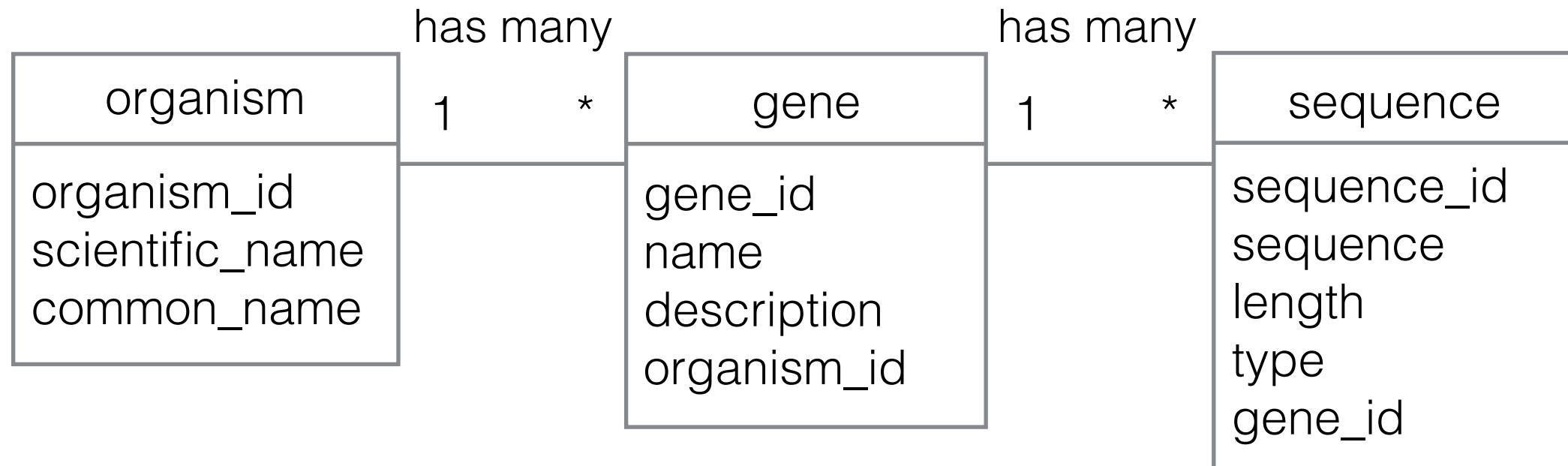


Exercises II

8. Use truncate to delete all the sequences from pineapple_demo (created in Exercises I)
9. Write in a file the code to create the next tables and load them into a database called ananas



Database Schema



```
psql -U postgres -d pineapple_demo -h localhost -a -f gene_tables.sql
```

```
psql -U postgres -d pineapple_demo -h localhost -a -f blastdb_tables.sql
```



Insert Rows

insert values into a table

```
INSERT INTO mytable (col1,col2) VALUES ('val1','val2');
```

```
INSERT INTO gene (name,description) VALUES ('Solyc01g001','unknown');
```

insert gene name and description into gene table



Update Rows

change column values from a row

change several column values from a row

```
UPDATE mytable SET col1='val1' WHERE col2='val2';
```

```
UPDATE mytable SET a=5, b=3, c=1 WHERE a>0;
```

```
UPDATE gene SET name = 'Solyc01g001' WHERE gene_id=1;
```

set gene name to Solyc01g001 for the row with gene_id 1



Delete Rows

delete the rows from table gene based on a condition

```
DELETE FROM mytable WHERE column='value';
```

```
DELETE FROM gene WHERE name='Solyc01g001';
```

delete gene named
Solyc01g001

```
DELETE FROM gene WHERE length<='100';
```

delete genes smaller than 100 nucleotides

gene
gene_id
name
description
length

Operator	Description
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
=	equal
<> or !=	not equal



Transactions

undo a transaction

```
BEGIN;  
UPDATE gene SET name='Solyc01g001' WHERE gene_id=1;  
ROLLBACK;
```

```
BEGIN;  
INSERT INTO gene (name,description) VALUES ('Solyc01g001','unknown');  
COMMIT;
```

commit a transaction

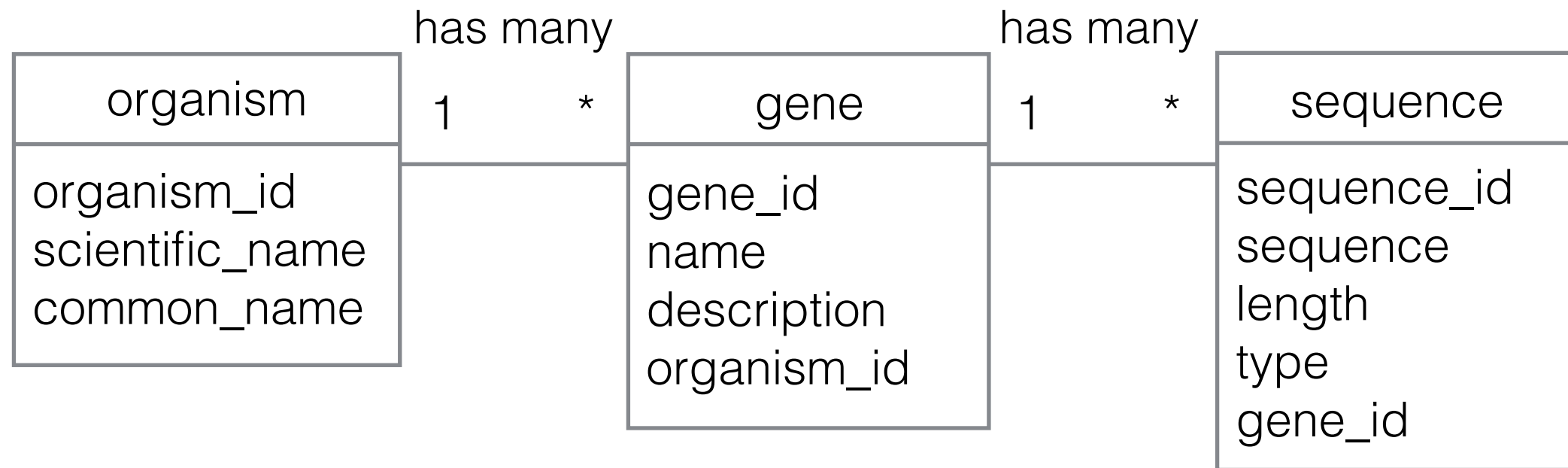


Exercises III

1. Insert into the tables organism, gene and sequence the first two sequences from the file pineapple_5k_proteins.fasta from the folder ~/Desktop/Data
2. Create a database called pineapple_copy and import the data from pineapple_backup.sql file created in *Exercises I*
3. Use transactions to delete the sequences longer than 100 amino acids from pineapple_copy



Database Schema



scripts to upload data. Eg: load_pineapple_fasta.pl





List database, schemas and tables		SQL query commands
\?	show help	SELECT * FROM mytable;
\h	SQL syntax help	SELECT column1 FROM mytable;
\h create table	SQL help for create table	SELECT ... WHERE mycolumn = 'value1';
\l	list databases	SELECT DISTINCT mycolumn from mytable;
\c mydb	connect to mydb database	SELECT ... GROUP BY mycolumn;
\dn	list schemas from a database	SELECT ... WHERE mycolumn IN (1,2);
\dt	list tables from public schema	SELECT ... WHERE mycolumn LIKE '%abc%';
\dt sgn.*	list tables from schema sgn	SELECT ... WHERE mycolumn ILIKE '%abc%';
\d blast_db	show blast_db table details	SELECT ... WHERE mycolumn IS NULL;
\q	quit psql	SELECT ... LIMIT 10;
		SELECT ... OFFSET 10;
		SELECT ... JOIN mytable using(common_column);

SELECT ... JOIN mytable on(common_column1=common_column2)

SQL development commands
CREATE DATABASE mydb;
ALTER DATABASE mydb RENAME TO newname;
DROP DATABASE mydb;
CREATE TABLE mytable;
ALTER TABLE mytable RENAME TO newname;
DROP TABLE mytable CASCADE;
ALTER TABLE mytable ADD COLUMN column_name column_type;
ALTER TABLE mytable RENAME COLUMN column_name TO newname;
ALTER TABLE mytable DROP COLUMN column_name RESTRICT;
INSERT INTO mytable (column1,column2) VALUES ('value1','value2');
UPDATE mytable SET column1='value1' WHERE column=value;
DELETE FROM mytable WHERE column='value';
begin;
commit;
rollback;

SQL query commands



Querying a Table

query all the data from the table

```
SELECT * FROM mytable;
```

```
SELECT name,description FROM gene;
```

get the name and description from all the genes

gene
gene_id
name
description
organism_id



Select - Where

query all the rows from a table for a condition

```
SELECT * FROM mytable WHERE col1='val1';
```

```
SELECT name,description FROM gene WHERE name='Aco000001';
```

get the name and description from gene Aco000001

gene
gene_id
name
description
organism_id



Limit and Offset

display the first 10 entries found

```
SELECT * FROM mytable LIMIT 10;
```

```
SELECT name FROM gene OFFSET 10;
```

display the last 10 gene names found in table gene



Select - Distinct

get the different entries for coll on mytable

```
SELECT DISTINCT coll FROM mytable;
```

```
SELECT DISTINCT description FROM gene;
```

get the different descriptions from the gene table

gene
gene_id
name
description
organism_id



Count

count rows from table

```
SELECT COUNT(*) FROM mytable;
```

```
SELECT COUNT(name) FROM gene;
```

count the gene names

```
SELECT COUNT(DISTINCT description) FROM gene;
```

count the different descriptions from gene table

gene
gene_id
name
description
organism_id



Sort Query Results

get all the rows from mytable sorted by column 1

ascendant order

```
SELECT * FROM mytable ORDER BY col1;
```

```
SELECT sequence_id,length FROM sequence ORDER BY length ASC;
```

```
SELECT sequence_id,length FROM sequence ORDER BY length DESC;
```

get all the sequence_id values sorted by sequence length in descendant order

sequence

sequence_id
sequence
length
type
gene_id



In

query all the rows for several values from a column

```
SELECT * FROM mytable WHERE col1 IN ('val1','val2');
```

```
SELECT * FROM gene WHERE gene_id IN (1,2);
```

get the rows for the genes with id 1 and 2

gene
gene_id
name
description
organism_id



Comparison Operators

In addition to the comparison operators, the special BETWEEN construct is available:

```
WHERE col_name BETWEEN x AND y
```

is equivalent to

```
WHERE col_name >= x AND col_name <= y
```

Notice that BETWEEN treats the endpoint values as included in the range. NOT BETWEEN does the opposite comparison:

```
WHERE col_name NOT BETWEEN x AND y
```

is equivalent to

```
WHERE col_name < x OR col_name > y
```

Operator	Description
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
=	equal
<> or !=	not equal



NULL

To check whether a value is or is not null, use the constructs:

```
expression IS NULL  
expression IS NOT NULL
```

or the equivalent, but nonstandard, constructs:

```
expression ISNULL  
expression NOTNULL
```



Like

return all the rows when col is equal to ribosomal

return all the rows with description not containing ribosomal

```
SELECT * FROM mytable WHERE col1 LIKE 'ribosomal';
```

```
SELECT * FROM gene WHERE description NOT LIKE '%ribosomal%';
```

```
SELECT * FROM gene WHERE description ILIKE 'ribosomal%';
```

return all the rows with description starting with ribosomal

case insensitive search



Like and iLike

LIKE	case sensitive comparison
iLIKE	case insensitive comparison
_	one character wildcard
%	wildcard

```
SELECT * FROM gene WHERE description LIKE 'ribosomal';
```

See the results and count the output for



- LIKE 'ribosomal'
- LIKE 'ribosomal%'
- LIKE '%ribosomal'
- LIKE '%ribosomal%'
- LIKE 'Ribosomal%'
- LIKE '_ribosomal%'
- iLIKE '%ribosomal%'
- NOT LIKE '%ribosomal'



Join Tables

join two tables by a common id

```
SELECT * FROM table1 JOIN table2 USING(common_id);
```

```
SELECT * FROM table1 JOIN table2 ON(id1=id2) where ...;
```

join two tables using two ids with equivalent values



Exercises IV

1. Find the 10 longest genes in pineapple_backup
2. Find the 10 shortest genes in pineapple_backup
3. How many different descriptions are in pineapple_backup
4. How many genes are in pineapple_backup
5. What is the description for the gene Aco005884?
6. What is the description for the gene with id 4341?
7. What is the gene with the first description alphabetically?
8. And the last one?



Exercises IV

9. How many gene description start with A (uppercase A)
10. How many different descriptions start with A
11. What is the name, description and length of the longest gene starting with A?
12. What are the genes with length between 118 and 120?
13. And their descriptions?
14. What is the longest protein containing ribosomal in its description? (case insensitive)
15. What are the genes smaller than 50 or longer than 3000 amino acids?



Exercises IV

16. What are the descriptions for the genes Aco000443.1 and Aco000775.1?
17. Save in a file all the organism, gene and sequence information for the entries with gene ids 427 and 752

