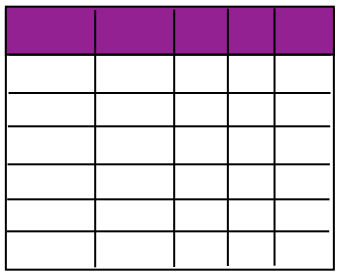
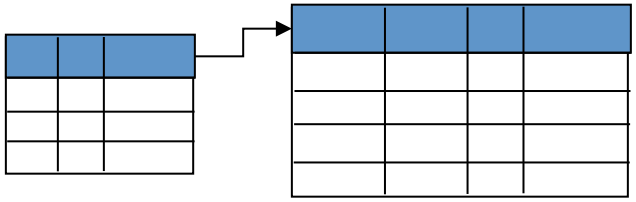
Key Idea: "Logical Data Independence"



views

```
SELECT *
  FROM my_sequences
```

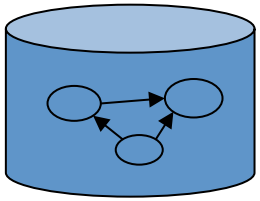
logical data independence



relations

```
SELECT seq
  FROM ncbi_sequences
 WHERE seq = 'GATTACGATATTA';
```

physical data independence



files and pointers

```
f = fopen('table_file');
fseek(10030440);
while (True) {
  fread(&buf, 1, 8192, f);
  if (buf == GATTACGATATTA) {
    . . .
```

What are Views?

- A view is just a query with a name
- We can use the view just like a real table

Why can we do this?

Because we know that every query returns a relation:
We say that the language is "algebraically closed"

View example

A view is a relation defined by a query

Purchase(customer, product, store)
Product(pname, price)

StorePrice(store, price)

```
CREATE VIEW  StorePrice AS
SELECT x.store, y.price
FROM  Purchase x, Product y
WHERE x.pid = y.pid
```

This is like a new table
StorePrice(store,price)

Customer(cid, name, city)
Purchase(customer, product, store)
Product(pname, price)

StorePrice(store, price)

How to Use a View?

- A "high end" store is a store that sold some product over 1000. For each customer, find all the high end stores that they visit. Return a set of (customer-name, high-end-store) pairs.

```
SELECT DISTINCT z.name, u.store
FROM Customer z, Purchase u, StorePrice v
WHERE z.cid = u.customer
AND u.store = v.store
AND v.price > 1000
```

Key Idea: Indexes

- Databases are especially, but not exclusively, effective at "Needle in Haystack" problems:
  - **Extracting small results from big datasets**
  - **Your query will always* finish, regardless of dataset size.**
  - **Indexes are easily built and automatically used when appropriate**

```
CREATE INDEX seq_idx ON sequence(seq);

SELECT seq
  FROM sequence
 WHERE seq = 'GATTACGATATTA';
```

*almost