Linearizing Tuples

Employees =

(23, Albert, 45000)
(77, Peter, 50000)
(42, Rob, 37000)

= 23, Albert, 45000, 42, Rob, 37000, 77, Peter, 50000
Employees = 

(23, Albert, 45000) 

(77, Peter, 50000) 

(42, Rob, 37000)
Row Stores

Employees =

(23, Albert, 45000)

→ (77, Peter, 50000)

(42, Rob, 37000)
Row Stores

Employees =

(23, Albert, 45000)
(77, Peter, 50000)
(42, Rob, 37000)
Employees = 

(23, Albert, 45000) 
(77, Peter, 50000) 
(42, Rob, 37000)
Column Stores

Employees =

(23, Albert, 45000)
(77, Peter, 50000)
(42, Rob, 37000)
Some Example Queries

```
SELECT  *  
FROM     employees
WHERE    name = 'Peter';
```

Employees =

(20, Albert, 45000)
(77, Peter, 50000)
(42, Rob, 37000)
Some Example Queries

```
SELECT * FROM employees
WHERE name = 'Peter';
```

```
SELECT * FROM employees;
```
Some Example Queries

```
SELECT *
FROM employees
WHERE name = 'Peter';

SELECT *
FROM employees;

INSERT INTO employees
WHERE (11, 'Jens', 'saarbrücken');
```
Some Example Queries

```
SELECT * 
FROM employees 
WHERE name = 'Peter';
```

```
SELECT * 
FROM employees;
```

```
INSERT INTO employees 
WHERE (11, 'Jens', 'saarbrücken');
```

```
DELETE FROM employees 
WHERE name = 'Rob';
```
Row Stores

Advantages:
- easy to implement
- good for transactional workloads
- good for single row access

Disadvantages:
- bad for analytics
- bad for wide tables

Employees =
- (20, Albert, 45000)
- (77, Peter, 50000)
- (42, Rob, 37000)
More Example Queries...

```
SELECT city_code
FROM employees
WHERE name = 'Peter';
```
More Example Queries...

```
SELECT city_code
FROM employees
WHERE name = 'Peter';
```

```
SELECT name, city_code
FROM employees;
```
More Example Queries...

SELECT city_code
FROM employees
WHERE name = 'Peter';

SELECT name, city_code
FROM employees;

ALTER TABLE employees
ADD lastname varchar;
More Example Queries...

```sql
SELECT city_code
FROM employees
WHERE name = 'Peter';

SELECT name, city_code
FROM employees;

ALTER TABLE employees
ADD lastname varchar;

ALTER TABLE employees
DROP lastname;
```

Employees =

(23, Albert, 45000)

(77, Peter, 50000)

(42, Rob, 37000)
More Example Queries...

```sql
SELECT city_code
FROM employees
WHERE name = 'Peter';

SELECT name, city_code
FROM employees;

ALTER TABLE employees
ADD lastname varchar;

ALTER TABLE employees
DROP lastname;

SELECT avg(city_code)
FROM employees;
```
Column Stores

Advantages:

- good for single to few column access
- good for analytical workloads

Disadvantages:

- bad when accessing multiple attributes
- (bad) for transactional workloads

Employees =

- (23, Albert, 45000)
- (77, Peter, 50000)
- (42, Rob, 37000)