شروع کار با SQL Server

درس پنجم: پیوند بین جدول‌ها و دستورات JOIN

سید کاوه احمدی

نکته

- مهمترین بخش در SQL جداول مرجع است.
- خروجی بند FROM یک جدول مرجع است که:
  - درجه‌ی (تعداد ستون‌های) آن برابر با مجموع درجه‌ی تمامی جداول مرجع ذکر شده در این بند است.
  - کاردینالیتی (تعداد سطرهای) آن برابر با ضرب دکارتی/کارتزین کاردینالیتی تمامی جداول مرجع ذکر شده در این بند است.
- تمامی بندهای دیگر روی جدول مرجع خروجی بند FROM اعمال می‌شود.
  - در یک دستور SQL، بند FROM اولین بندی است که اجرا می‌شود.

نکته

```
SELECT
        T1.*, T2.*
FROM
        T1, T2
```

به وسیله « . » مشخص می‌نماییم که فیلد یا فیلدها از کدام جدول مرجع مورد نظر است

A	B	C
a	b	c
k	l	m
m	n	o

T1

D	C	W
d	e	f
g	h	l

T2

A	B	T_1.C	D	T_2.C	W
a	b	c	d	e	f
a	b	c	g	h	l
k	l	m	d	e	f
k	l	m	g	h	l
m	n	o	d	e	f
m	n	o	g	h	l

استفاده از ضرب دکارتی در اتصال جدول‌ها

- عنوان خبر و نام نویسنده اخبار را می‌خواهیم:

```
SELECT
      n.Title, a.Name
FROM
      News AS n, Authors AS a
WHERE
      n.AuthorId = a.Id
```

استفاده از ضرب دکارتی در اتصال جدول‌ها

- در پرسجوی زیر اخباری که دسته‌بندی ندارند به دلیل برقرار نبودن شرط در خروجی ظاهر نمی‌شوند:

```
SELECT
      n.Title, c.Name
FROM
      News n, Categories c
WHERE
      n.CategoryId = c.Id
```

نکته

- تعداد جدول‌های قابل استفاده در بند FROM نامحدود است.

```
SELECT
      T1.Title, Table2.Name, T3.Name
FROM
      Table1 T1, Table2, Table3 T3
WHERE
      T1.Id = T3.Id
```

نکته

- زمانی که از یک اسم مستعار برای یک جدول استفاده نماییم، جدول مرجع صرفا به همان اسم مستعار شناخته خواهد شد.
- در مورد اسامی هیچ ابهامی نباید وجود داشته باشد.

```
SELECT
        A,
        Table1.b,
        C
FROM
        Table1 T1, Table2 T2
```

درست: در جداول مرجع فقط یک فیلد A داریم.

نادرست: جدول مرجع با نام T1 شناخته می‌شود.

نادرست: کدام C؟ (در هر دو جدول مرجه فیلد C وجود دارد)

A	B	C
a	b	c
k	l	m
m	n	o

Table1

D	C	W
d	e	f
g	h	l

Table2

SQL Join

- تا حالا دیدیم که برای پیوند دو جدول می‌توان از پرسش‌های فرعی و ضرب دکارتی بین جدول‌ها استفاده کرد.
- روش دیگری برای این منظور نیز وجود دارد که ناظر بر عملگر پیوند در جبر رابطه‌ای است.

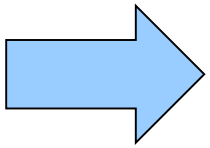
Inner Join

id	name
1	Pirate
2	Monkey
3	Ninja
4	Spaghetti

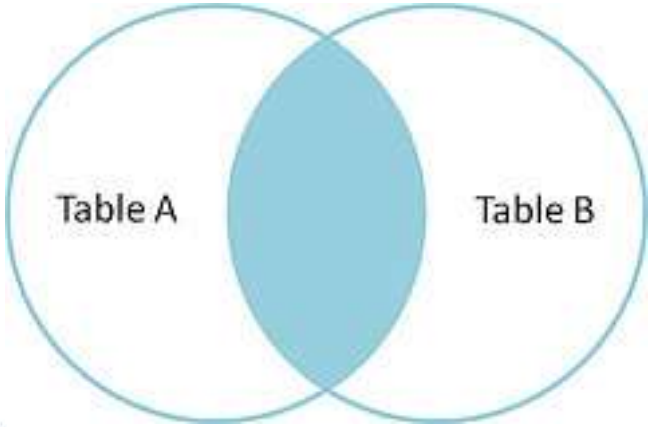
TableA

id	name
1	Rutabaga
2	Pirate
3	Darth Vader
4	Ninja

TableB



id	name	id	name
1	Pirate	2	Pirate
3	Ninja	4	Ninja




```
SELECT * FROM TableA
INNER JOIN TableB
ON TableA.name = TableB.name
```

Inner Join

- پرسجویی بنویسید که عنوان و نام نویسنده اخبار را بازگرداند:
  - ذکر INNER اختیاری است.

```
SELECT
	n.Title,
	a.Name
FROM
	News n
INNER JOIN
	Authors a
ON
	n.CategoryId = a.Id
```

```
SELECT
	n.Title,
	a.Name
FROM
	News n
JOIN
	 Authors a
ON
	n.CategoryId = a.Id
```

سوال

- برای نوشتن پرسجویی برای بازگرداندن عنوان و نام دسته‌ی اخبار، از آنجایی که فقط اشتراک دو جدول ذکر می‌شود و دسته `null` در جدول `Categories` وجود ندارد، با استفاده از `Inner Join` خبرهایی که `CategoryId` آنها `NULL` است در نتیجه پرسجو نمی‌آیند

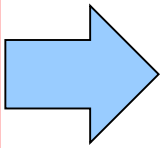
Left/Right Outer Join

id	name
1	Pirate
2	Monkey
3	Ninja
4	Spaghetti

TableA

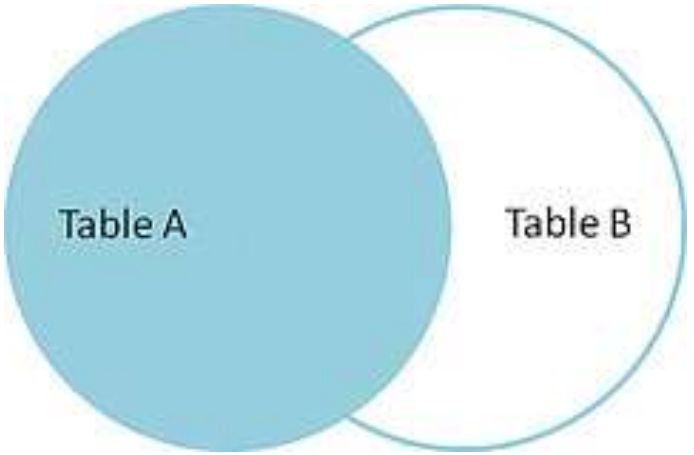
id	name
1	Rutabaga
2	Pirate
3	Darth Vader
4	Ninja

TableB



id	name	id	name
1	Pirate	2	Pirate
2	Monkey	null	null
3	Ninja	4	Ninja
4	Spaghetti	null	null

```
SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.name = TableB.name
```



Left/Right Outer Join

- پرسجویی بنویسید که عنوان و نام دسته‌ی اخبار را بازگرداند:
  - حل مسئله کلیدهای خارجی NULL با استفاده از Join از طرفین.
  - ذکر Outer اختیاری است.

```
SELECT
	n.Title, c.Name
FROM
	News n
LEFT OUTER JOIN
	Categories c
ON
	n.AuthorId = c.Id
```

```
SELECT
	n.Title, c.Name
FROM
	Categories c
RIGHT OUTER JOIN
	News n
ON
	n.AuthorId = c.Id
```

Left/Right Outer Join

- پرسجویی بنویسید که عنوان، نام نویسنده و نام دسته‌ی اخبار را بازگرداند:
  - `CategoryId` می‌توانست `NULL` باشد ولی `AuthorId` خیر!

```
SELECT
	n.Title, c.Name, a.Name
FROM
	News n
LEFT JOIN
	Categories c
ON
	n.CategoryID = c.ID
JOIN
	Authors a
ON
	n.AuthorID = a.ID
```

Left/Right Outer Join

- پرسجویی بنویسید که عنوان، نام نویسنده و نام دسته‌ی اخبار را بازگرداند:
  - `CategoryId` می‌توانست `NULL` باشد ولی `AuthorId` خیر!
  - ممکن است بعدا نظرمان در مورد `AuthorId` هم عوض شود:

```
SELECT
	n.Title, c.Name, a.Name
FROM
	News n
LEFT JOIN
	Categories c
ON
	n.CategoryID = c.ID
LEFT JOIN
	Authors a
ON
	n.AuthorID = a.ID
```

مثال

- پرسجویی بنویسید که عنوان، نام نویسنده، نام دسته‌ی اخبار و تعداد نظرات آنرا بازگرداند:

```
SELECT
        n.Title, c.Name, a.Name, COUNT(co.Id)
FROM
        News n
LEFT JOIN
        Categories c
ON
        n.CategoryID = c.ID
JOIN
        Authors a
ON
        n.AuthorID = a.ID
LEFT JOIN
        Comments co
ON
        n.Id = co.NewsId
GROUP BY
        n.Title, c.Name, a.Name
```

همانطور که پیش از این اشاره کردیم، هنگام استفاده از توابع تجمعی، GROUP BY باید براساس تمام ستون‌های انتخاب شده که تابع تجمعی نیستند انجام شود.

مثال

- روش دوم:

```
SELECT
n.Title, c.Name, a.Name, (SELECT COUNT(*)
                          FROM Comments co
                          WHERE n.Id = co.NewsId)
FROM
        News n
LEFT JOIN
        Categories c
ON
        n.CategoryID = c.ID
JOIN
        Authors a
ON
        n.AuthorID = a.ID
```

مثال

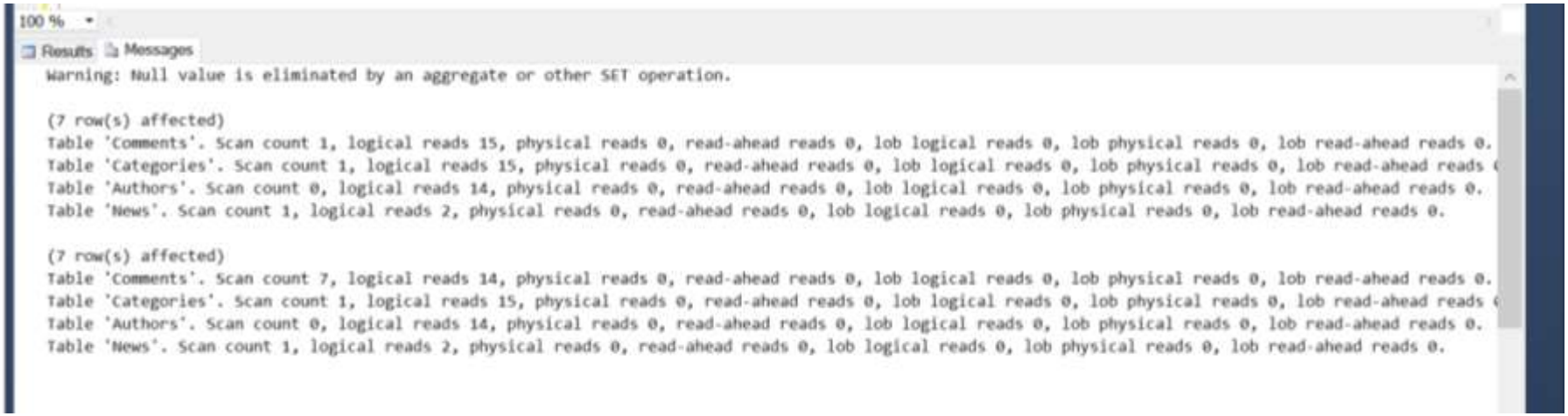
- کدام بهتر است؟
  - درحالت کلی پاسخ روشنی برای این پرسش وجود ندارد.
  - در بسیاری از موارد SQL Optimizer بهترین شکل پرسجو را ایجاد می‌کند و آنرا اجرا می‌کند (مثلا در بسیاری از موارد ضرب دکارتی جداول را به JOIN تبدیل می‌کند).
  - در حالت کلی JOIN بهتر از ضرب دکارتی است. دست کم به خاطر اجباری بودن بند ON، احتمال اشتباه در پیوند جدول‌ها را کاهش می‌دهد. البته استفاده از توابع تجمعی، GROUP BY، ORDER BY، DISTINCT و... می‌تواند قواعد دیگری را رقم بزند.
  - بهتری روش مقایسه عملکرد SQL Server به وسیله Execution Plan است.

مثال

- پرسجوها:

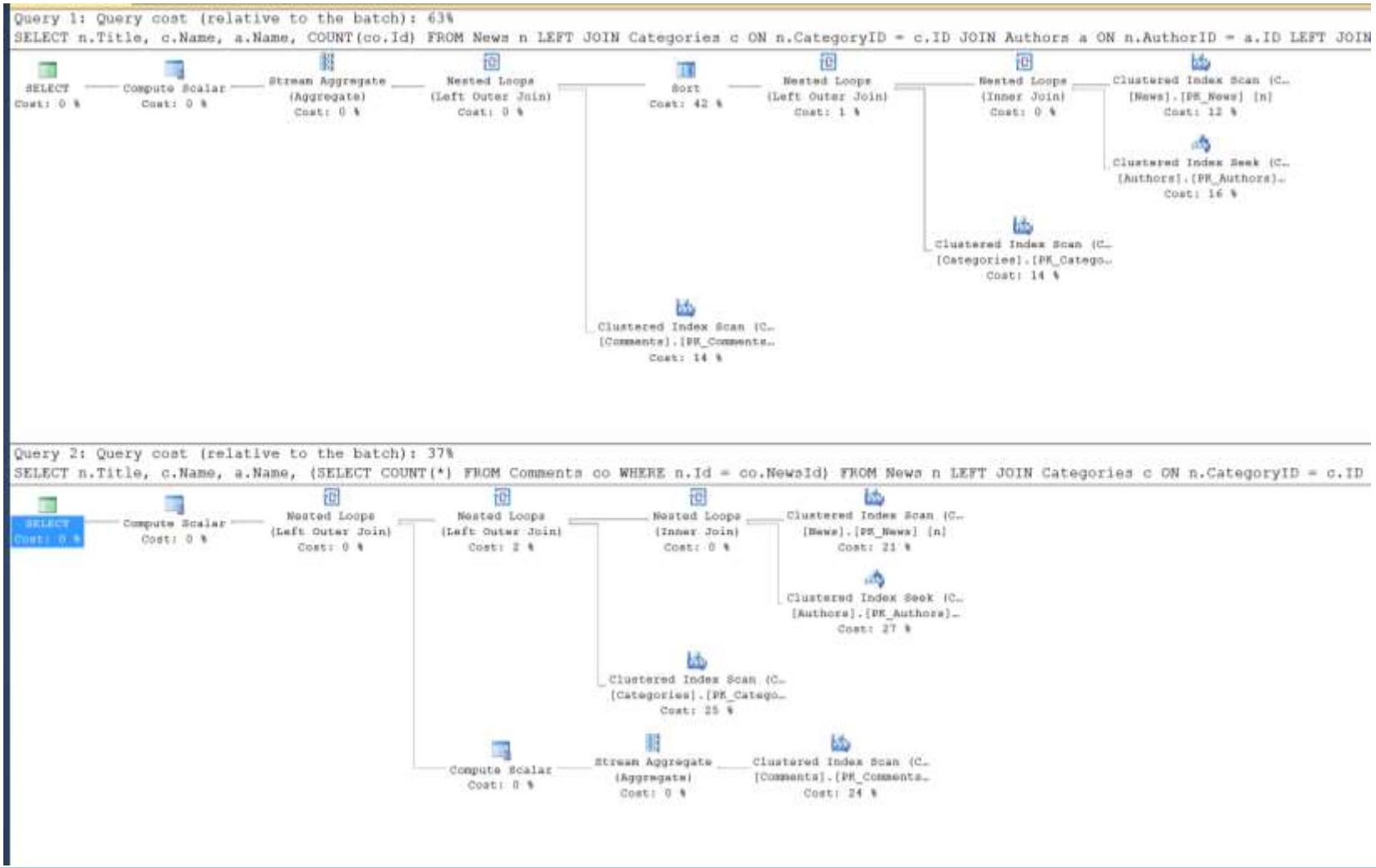
مثال

- IO Statics



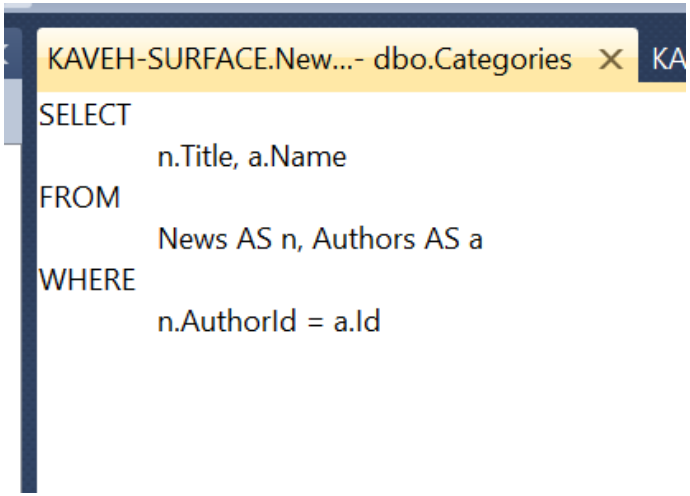
مثال

- Execution Plan

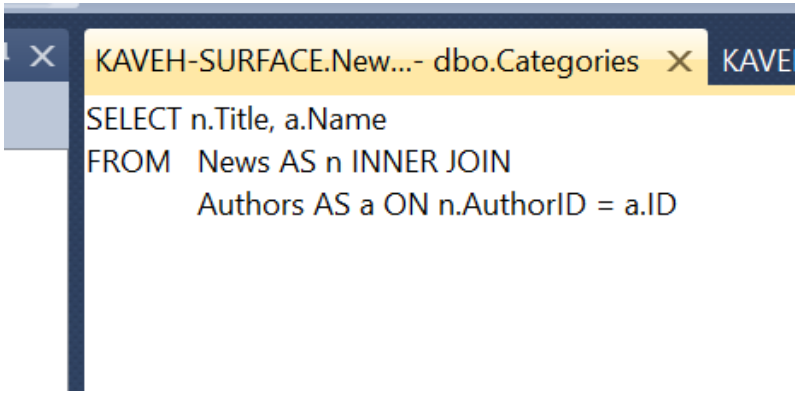


مثال

- SQL Optimizer پیش از اجرای پرسجو آنرا بهینه می‌کند (به خیال خودش!)
  - در چنین شرایطی Execution Plan هر دو پرسجو یکسان نمایش داده می‌شود (چون عملا پرسجوی یکسانی اجرا شده است).



چیزی که اجرا می‌کنم



چیزی که اجرا می‌شود

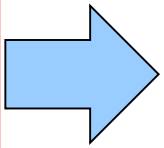
Left/Right Outer Join

id	name
1	Pirate
2	Monkey
3	Ninja
4	Spaghetti

TableA

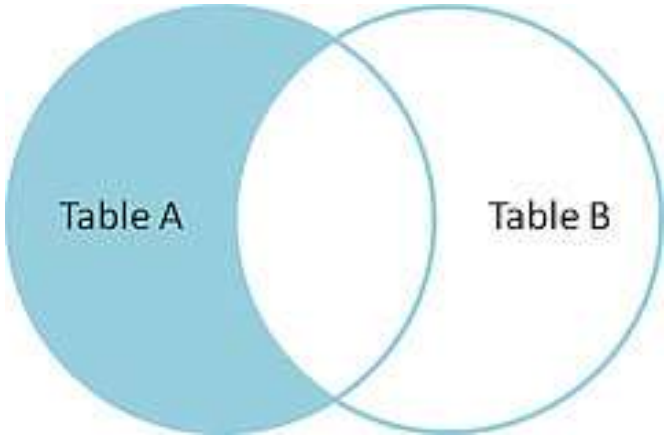
id	name
1	Rutabaga
2	Pirate
3	Darth Vader
4	Ninja

TableB



id	name	id	name
2	Monkey	null	null
4	Spaghetti	null	null

```
SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.name = TableB.name
WHERE TableB.id IS null
```



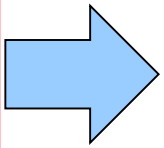
Full Outer Join

id	name
1	Pirate
2	Monkey
3	Ninja
4	Spaghetti

TableA

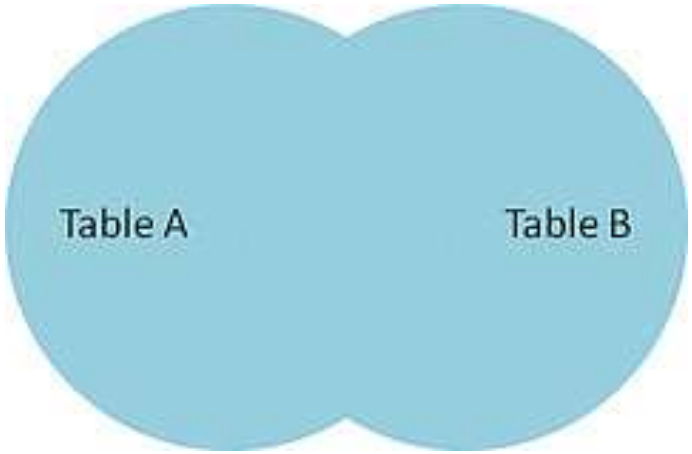
id	name
1	Rutabaga
2	Pirate
3	Darth Vader
4	Ninja

TableB



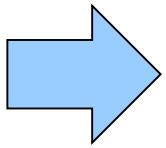
id	name	id	name
1	Pirate	2	Pirate
2	Monkey	null	null
3	Ninja	4	Ninja
4	Spaghetti	null	null
null	null	1	Rutabaga
null	null	3	Darth Vader

```
SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.name = TableB.name
```



Cross Join

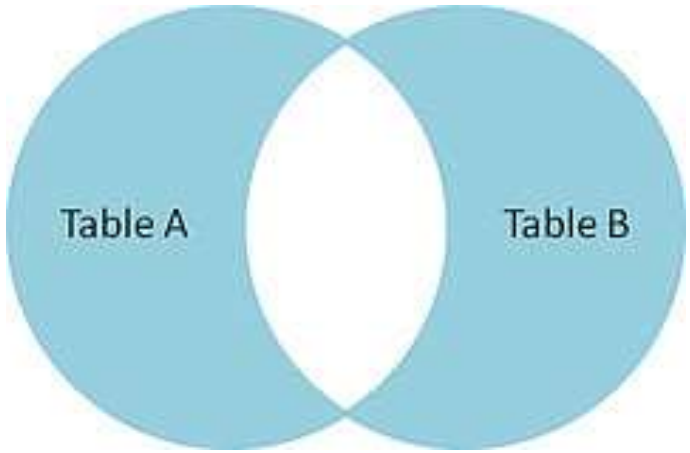
id	name	id	name
1	Pirate	1	Rutabaga
2	Monkey	2	Pirate
3	Ninja	3	Darth Vader
4	Spaghetti	4	Ninja



id	name	id	name
2	Monkey	null	null
4	Spaghetti	null	null
null	null	1	Rutabag
null	null	3	Darth V

```
SELECT * FROM TableA
CROSS JOIN TableB
```

```
SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.name = TableB.name
WHERE TableA.id IS null
OR TableB.id IS null
```



سایر مطالب مرتبط

- ANY, ALL, EXISTS
- Nested Loop VS. Merge Join VS. Hash Join
- مراحل اجرای یک پرسجو در SQL Server