Tabelle1

Hinweise

Ergebnis (0: nicht äquivalent, 1: äquivalent, (pm=0..1): tw. Äquivalent, error: Fehler → System kann Querys nicht vergleichen)

pm = partial marking (z.B. zu 85% korrekt)

pm, sonst 0 → falls Verfahren partielle Bewertung unterstützt wird pm erwartet, ansonsten nicht äquivalent erwartet

Muster (Name = x, pm = true/false)

tats. = tatsächlich

				Ergebnis		
Mutations-Kategorie/ SQL-Feature	Nr.	Query (Musterlösung)	Test-Query (Kandidat)	erwartet	tats.	Anmerkung
columns select vs. Select *	1	SELECT * FROM Album;	SELECT AlbumId, Title, ArtistId FROM Album;	1		
	2	SELECT AlbumId, Title, ArtistId FROM Album;	SELECT * FROM Album;	1		
where: same semantic, minimal syntax change	3	SELECT TrackId FROM PlaylistTrack WHERE TrackId < 100;	SELECT TrackId FROM PlaylistTrack WHERE TrackId <= 99;	1		
Semicolon ;	4	SELECT Name FROM Genre;	SELECT Name FROM Genre	1		
syntax sql keywords (upper, lower case)	5	SELECT Name FROM MediaType;	select Name FrOm MediaType;	1		
column count differing	6	SELECT InvoiceDate, BillingAddress, BillingCity FROM Invoice;	SELECT BillingAddress, BillingCity FROM Invoice;	pm, sonst 0		
	7	SELECT BillingAddress, BillingCity FROM Invoice;	SELECT InvoiceDate, BillingAddress, BillingCity FROM Invoice;	pm, sonst 0		
column ordering	8		SELECT BillingCity, BillingAddress FROM Invoice;	pm, sonst 0		
order by	9	SELECT TrackId, Name, GenreId from Track ORDER BY GenreId ASC;	SELECT TrackId, Name, GenreId from Track ORDER BY GenreId;	1		
	10	SELECT TrackId, Name, GenreId from Track ORDER BY GenreId DESC;	SELECT TrackId, Name, GenreId from Track;	pm, sonst 0		
	11		SELECT TrackId, Name, GenreId from Track ORDER BY GenreId ASC;	pm, sonst 0		
	12	SELECT Name, MediaTypeId FROM Track	SELECT Name, MediaTypeId FROM Track WHERE MediaTypeId > 2 and MediaTypeId > 1;	1		

Redundanzen	13	WHERE MediaTypeId > 2;	SELECT Name, MediaTypeId FROM Track WHERE MediaTypeId > 2 or MediaTypeId > 1;	pm, sonst 0		
Sub-Query/Join	14	select Track.TrackId, Track.Name, Track.Composer from Track join (select MediaType.MediaTypeId, MediaType.Name from MediaType) as t_MediaType ON Track.MediaTypeId = t_MediaType.MediaTypeId;	select Track.TrackId, Track.Name, Track.Composer from Track join (select MediaType.MediaTypeId, MediaType.Name from MediaType) as t_MediaType ON Track.MediaTypeId = t_MediaType.MediaTypeId;	pm, sonst 0		
	15		select Track.TrackId, Track.Name, Track.Composer from Track Join MediaType ON Track.MediaTypeId = MediaType.MediaTypeId;	pm, sonst 0		
	16		select Track.TrackId, Track.Name, Track.Composer from Track join MediaType ON Track.MediaTypeId = MediaType.MediaTypeId WHERE MediaType.Name = 'AAC audio file';	1		
	17		select Track.TrackId, Track.Name, Track.Composer from Track join MediaType ON Track.MediaTypeId = MediaType.MediaTypeId AND MediaType.Name = 'AAC audio file';	1		

	18		select Track.TrackId, Track.Name, Track.Composer from Track, MediaType where Track.MediaTypeId = MediaType.MediaTypeId and (MediaType.Name = 'AAC audio file')	1		
distinct	19	SELECT DISTINCT PlaylistId FROM PlaylistTrack;	SELECT PlaylistId FROM PlaylistTrack;	pm, sonst 0		
where, and	20	SELECT * FROM Employee WHERE Title = 'Sales Support Agent';	SELECT * FROM Employee;	pm, sonst 0		keine Where-Klausel
	21		SELECT * FROM Employee WHERE ReportsTo = 2;	pm, sonst 0		zufällig gleiches Ergebnis-Set aber andere Syntax & Semantik
	22	SELECT Email FROM Employee WHERE City = 'Calgary' AND ReportsTo = 1;	SELECT Email FROM Employee WHERE City = 'Calgary';	pm, sonst 0		
or	23	SELECT LastName, FirstName, City FROM Employee WHERE City = 'Edmonton' OR City = 'Lethbridge';	SELECT LastName, FirstName, City FROM Employee WHERE City = 'Edmonton';	pm, sonst 0		
	24		SELECT LastName, FirstName, City FROM Employee WHERE City = 'Edmonton' OR City = 'Calgary';	pm, sonst 0		
not	25	SELECT LastName, FirstName, City FROM Employee WHERE NOT City = 'Calgary';	SELECT LastName, FirstName, City FROM Employee WHERE City = 'Lethbridge' OR City = 'Edmonton';	pm, sonst 0		Semantik → Test für Verfahren mit pragmatischen Ansatz
	26		SELECT LastName, FirstName, City FROM Employee WHERE NOT City = 'Edmonton';	pm, sonst 0		
limit	27	SELECT LastName, FirstName, City FROM Employee LIMIT 3;	SELECT LastName, FirstName, City FROM Employee LIMIT 4;	pm, sonst 0		
	28		SELECT LastName, FirstName, City FROM Employee;	0		

Tabelle1

min, max	29	SELECT MIN(UnitPrice) FROM InvoiceLine;	SELECT UnitPrice FROM InvoiceLine LIMIT 1;	0		Verfahren mit pragmatischen Ansatz
	30		SELECT MAX(UnitPrice) FROM InvoiceLine;	0		
	31	SELECT MAX(UnitPrice) FROM InvoiceLine;	SELECT UnitPrice FROM InvoiceLine LIMIT 1;	0		
count	32	SELECT COUNT(UnitPrice) FROM InvoiceLine WHERE UnitPrice = 0.99;	SELECT UnitPrice FROM InvoiceLine WHERE UnitPrice = 0.99;	0		
avg	33	SELECT avg(Total) FROM Invoice;	SELECT Total FROM Invoice;	0		
sum	34	SELECT sum(Total) FROM Invoice;		0		
like	35	SELECT ArtistId, Name FROM Artist WHERE Name LIKE '%Metal%';	SELECT ArtistId, Name FROM Artist;	0		
	36	SELECT ArtistId, Name FROM Artist WHERE Name LIKE 'Sant%';	SELECT ArtistId, Name FROM Artist WHERE Name LIKE 'San%';	pm, sonst 0		
	37	SELECT ArtistId, Name FROM Artist WHERE Name LIKE 'M%o';	SELECT ArtistId, Name FROM Artist WHERE Name LIKE 'M%';	pm, sonst 0		
	38	SELECT ArtistId, Name FROM Artist WHERE Name LIKE '_ %';	SELECT ArtistId, Name FROM Artist WHERE Name LIKE '_%';	pm, sonst 0		
in	39	SELECT InvoiceId, InvoiceDate, BillingCountry FROM Invoice WHERE BillingCountry IN ('USA', 'Canada', 'Brazil');	SELECT InvoiceId, InvoiceDate, BillingCountry FROM Invoice;	0		
	40		SELECT InvoiceId, InvoiceDate, BillingCountry FROM Invoice WHERE BillingCountry IN ('USA', 'Brazil');	pm, sonst 0		
between	41	SELECT InvoiceId, InvoiceDate, BillingCountry FROM Invoice WHERE InvoiceDate between '2012-06-04 00:00:00' AND '2012-07-28 00:00:00';	SELECT InvoiceId, InvoiceDate, BillingCountry FROM Invoice;	0		
	42		SELECT InvoiceId, InvoiceDate, BillingCountry FROM Invoice WHERE InvoiceDate between '2012-05-04 00:00:00' AND '2012-07-28 00:00:00';	pm, sonst 0		
	43	SELECT InvoiceId, CustomerId FROM Invoice WHERE InvoiceId between 50 AND 100;	SELECT InvoiceId, CustomerId FROM Invoice WHERE InvoiceId between 0 AND 120;	pm, sonst 0		
column alias	44	SELECT Name AS Genres FROM Genre;	SELECT Name AS Genre FROM Genre;	pm, sonst 0		

union	45	SELECT City FROM Customer UNION SELECT City FROM Employee;	SELECT City FROM Customer;	0		
	46		SELECT City FROM Customer UNION SELECT BillingCity FROM Invoice;	pm, sonst 0		
	47	SELECT City FROM Customer UNION ALL SELECT City FROM Employee;	SELECT City FROM Customer UNION SELECT City FROM Employee;	pm, sonst 0		
group by, having	48	SELECT InvoiceId, CustomerId, count(CustomerId) FROM Invoice GROUP BY CustomerId;	SELECT InvoiceId, CustomerId, count(CustomerId) FROM Invoice;	0		
	49		SELECT InvoiceId, CustomerId, count(CustomerId) FROM Invoice GROUP BY InvoiceId;	pm, sonst 0		
	50	SELECT CustomerId, count(CustomerId) FROM Invoice group by CustomerId having count(CustomerId) < 7;	SELECT CustomerId, count(CustomerId) FROM Invoice group by CustomerId;	0		
	51		SELECT CustomerId, count(CustomerId) FROM Invoice group by CustomerId having count(CustomerId) > 1;	pm, sonst 0		
exists	52	SELECT TrackID, Name, GenreID FROM Track WHERE EXISTS (SELECT GenreID FROM Genre WHERE Track.GenreId = Genre.GenreId AND (Genre.Name = 'Reggae' OR Genre.Name = 'Jazz'));	SELECT TrackID, Name, GenreID FROM Track WHERE EXISTS (SELECT GenreID FROM Genre WHERE Track.GenreId = Genre.GenreId AND Genre.Name = 'Reggae' OR Genre.Name = 'Jazz');	pm, sonst 0		
	53		SELECT TrackID, Name, GenreID FROM Track;	0		
	54		SELECT TrackID, Name, GenreID FROM Track WHERE GenreID IN (SELECT GenreID FROM Genre WHERE Track.GenreId = Genre.GenreId AND (Genre.Name = 'Reggae' OR Genre.Name = 'Jazz'));	1		

Inner, left, self join	55	SELECT PlaylistTrack.PlaylistId, Track.TrackId, Track.Name, Genre.Name FROM ((PlaylistTrack INNER JOIN Track ON PlaylistTrack.TrackId = Track.TrackId) INNER join Genre ON Genre.GenreId = Track.GenreId);	SELECT PlaylistTrack.PlaylistId, Track.TrackId, Track.Name, Genre.Name FROM ((PlaylistTrack JOIN Track ON PlaylistTrack.TrackId = Track.TrackId) INNER join Genre ON Track.GenreId = Genre.GenreId);	1		
	56	SELECT Customer.CustomerId, Invoice.InvoiceId FROM Customer LEFT JOIN Invoice ON Customer.CustomerId = Invoice.InvoiceId	SELECT Customer.CustomerId, Invoice.InvoiceId FROM Customer LEFT OUTER JOIN Invoice ON Customer.CustomerId = Invoice.InvoiceId	1		
	57	SELECT A.FirstName AS CustomerName1, B.FirstName AS CustomerName2, A.City FROM Customer A, Customer B WHERE A.CustomerID != B.CustomerID AND A.City = B.City ORDER BY A.City;	SELECT X.FirstName AS CustomerName1, Y.FirstName AS CustomerName2, Y.City FROM Customer X, Customer Y WHERE Y.CustomerID <> X.CustomerID AND X.City = Y.City ORDER BY Y.City;	1		
to much joins	58	SELECT PlaylistTrack.PlaylistId, Track.TrackId, Track.Name FROM (PlaylistTrack INNER JOIN Track ON PlaylistTrack.TrackId = Track.TrackId);	SELECT PlaylistTrack.PlaylistId, Track.TrackId, Track.Name, Genre.Name FROM ((PlaylistTrack INNER JOIN Track ON PlaylistTrack.TrackId = Track.TrackId) INNER join Genre ON Genre.GenreId = Track.GenreId);	0		